A **Capstone** Project report submitted

in partial fulfillment of requirement for the award of degree

**BACHELOR OF TECHNOLOGY**

in

**SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE**

by

**2203A52161 MITTA DHEERAJ**

Under the guidance of

**Dr.Ramesh Dadi**

Assistant Professor, School of CS&AI.



SR University, Ananthsagar,Warangal,Telagnana-506371

# SR University

Ananthasagar, Warangal.



## CERTIFICATE

This is to certify that this project entitled “**DATA ANALYSIS USING PYTHON**” is the bonafied work carried out **MITTA DHEERAJ** as a Major Project for the partial fulfillment to award the degree BACHELOR OF TECHNOLOGY in School of Computer Science and Artificial Intelligence during the academic year 2024-2025 under our guidance and Supervision.

### Dr.Ramesh Dadi Dr. M.Sheshikala

Assistant Professor Professor & Head,

SR University School of CS&AI,

Anathasagar,Warangal SR University

Ananthasagar, Warangal.

**Reviewer-1 Reviewer-2**

Name: Name:

Designation: Designation:

Signature: Signature:

# DATASET

**Project -1**

"Personalized Health Fitness Tracker," which aims to analyse a dataset containing health and fitness metrics to potentially predict or classify fitness-related outcomes. The dataset includes features such as age, gender, weight, height, heart rate metrics, workout type, and calories burned, among others. The notebook employs Python libraries like pandas, matplotlib, seaborn, and scikit-learn to perform data exploration, preprocessing, and model evaluation. The code snippet also includes a performance comparison of multiple machine learning models (XGBoost, Random Forest, SVM, and Decision Tree) using metrics like accuracy, precision, recall, and F1-score, visualized through bar plots. The primary goal appears to be developing a predictive model for personalized fitness tracking, leveraging the dataset's rich features.

**Project – 2**

"Chess Master Style Classifier" is a data science project aimed at analyzing chess game data to classify or predict playing styles, strategies, or outcomes. The dataset (unzipped from Chess.zip) typically includes game records such as player names, moves, outcomes, and ELO ratings. The goal is to leverage historical game patterns to detect strategies of players or determine win probabilities based on opening moves. The project might use libraries such as pandas, numpy, and scikit-learn, alongside domain-specific libraries like python-chess for board representation. Through data cleaning, move parsing, and game feature extraction, models can be trained for classification (e.g., opening type or player style) or regression (e.g., predicting ELO improvement). The project also supports visualizations of move frequencies and heatmaps of board states, making it a compelling application of AI in board game analytics.

**Project – 3**

# 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.

# 

# METHODOLOGY

**Project – 1**

**Dataset Preparation**: The dataset, high\_popularity\_spotify\_data.csv, contains over 1100 rows and 19 columns featuring numerical and categorical audio metrics from Spotify tracks. Key features include danceability, energy, acousticness, tempo, duration\_ms, valence, and popularity. Initial inspection using data.head(), data.shape, data.describe(), and data.info() helps reveal dataset structure, statistical spread, and absence of null values.

**Data Preprocessing:** The popularity feature is transformed into a binary classification target—labeling songs as "high popularity" if their score is above a certain threshold (e.g., 70). The dataset is then normalized using StandardScaler() to ensure uniform scaling of audio metrics. Categorical features, if present (e.g., key, mode), are label-encoded or one-hot encoded for model compatibility.

**Feature Extraction**: Features like danceability, energy, valence, and tempo are selected based on correlation with popularity, while highly correlated or redundant attributes are removed through pairwise correlation analysis. Dimensionality reduction via PCA is optionally applied for exploratory purposes but not retained in final model training to preserve feature interpretability.

**Model Architecture**: Multiple machine learning classifiers—Logistic Regression, Random Forest, XGBoost, and Support Vector Machine—are employed for classification. Each model is implemented using scikit-learn and xgboost, with hyperparameters tuned through GridSearchCV or manual testing to optimize performance on audio feature space.

**Model Training**: The models are trained using an 80/20 train-test split, with train\_test\_split ensuring reproducibility and class distribution. Each classifier is fit on the scaled features and evaluated on the test set using model.fit() and model.predict().

**Performance Evaluation:** Accuracy, precision, recall, and F1-score are calculated for all models, with weighted averages used due to potential class imbalance. Performance is visualized using bar plots and confusion matrices via matplotlib and seaborn, enabling a comparative analysis to identify the most effective classifier for predicting song popularity.

**Project -2**

**Dataset Preparation:** The dataset consists of competitive chess games with each record including attributes like White, Black, WhiteElo, BlackElo, Opening, VictoryStatus, and TimeControl. The dataset is loaded using pandas, and data.head(), data.shape, data.info(), and data.describe() are used to understand structure, dimensions, and basic statistics. Duplicate entries are dropped and data types are verified for consistency.

**Data Preprocessing:** Categorical attributes such as Opening, VictoryStatus, and TimeControl are encoded using Label Encoding and One-Hot Encoding as needed. Player Elo ratings are converted to numerical format for both white and black players, and new features like EloDifference = WhiteElo - BlackElo are engineered to aid model prediction. Missing values, if any, are addressed via imputation or row removal.

**Feature Extraction:** Key features used for modeling include WhiteElo, BlackElo, EloDifference, Opening, and TimeControl. The target variable is VictoryStatus, which is encoded into classes like win, resign, timeout, or draw. Feature selection is informed by correlation heatmaps and domain intuition regarding strategic significance in chess outcomes.

**Model Architecture:** Machine learning models such as Random Forest, Decision Tree, Logistic Regression, and XGBoost are implemented using scikit-learn and xgboost. These models are chosen for their ability to handle mixed feature types and provide interpretability or robustness. Hyperparameters like max\_depth, n\_estimators, and learning\_rate are tuned for optimal performance.

**Model Training:** An 80/20 split is used to partition the dataset into training and test sets. Models are trained using model.fit() and evaluated on unseen test data. StratifiedShuffleSplit or cross\_val\_score is optionally applied to maintain class distribution during training and validate robustness.

**Performance Evaluation:** Accuracy, precision, recall, and F1-score are computed for each classifier. Performance is visualized via confusion matrices and bar charts using seaborn and matplotlib. The evaluation emphasizes balanced performance across multiple outcome classes to determine the best model for predicting chess match results.

**Project – 3**

**Dataset Preparation**: The dataset used in this project is an integrated set combining elements from multiple sources to explore multi-domain classification or prediction tasks. The dataset is loaded and inspected using pandas with functions such as data.head(), data.shape, and data.info() to understand the data structure and detect missing or anomalous values. The dataset includes a mix of audio features, game metadata, and performance metrics, with over 1000 rows and several feature columns spanning different domains.

**Data Preprocessing:** Null values are identified and handled using imputation techniques—mean for numerical and mode for categorical variables. All categorical features are label encoded or one-hot encoded based on model requirements. Numerical features are standardized using StandardScaler() to ensure uniform scale, and features are renamed or engineered for clarity (e.g., merging common attributes, deriving new variables like engagement score or combined rating).

**Feature Extraction:** Key features are extracted based on their relevance to the combined prediction task—such as predicting the likelihood of success or outcome based on cross-domain inputs (e.g., tempo, elo ratings, energy levels, time control, etc.). Correlation analysis is performed to select features that significantly contribute to the target variable, and feature redundancy is reduced by removing multicollinear features.

**Model Architecture**: A suite of classification models including Logistic Regression, Random Forest, XGBoost, and SVM is constructed using scikit-learn and xgboost. These models are selected for their proven performance in mixed-type data and ability to handle nonlinear relationships. Grid search or manual tuning is used to optimize hyperparameters like C, n\_estimators, and kernel.

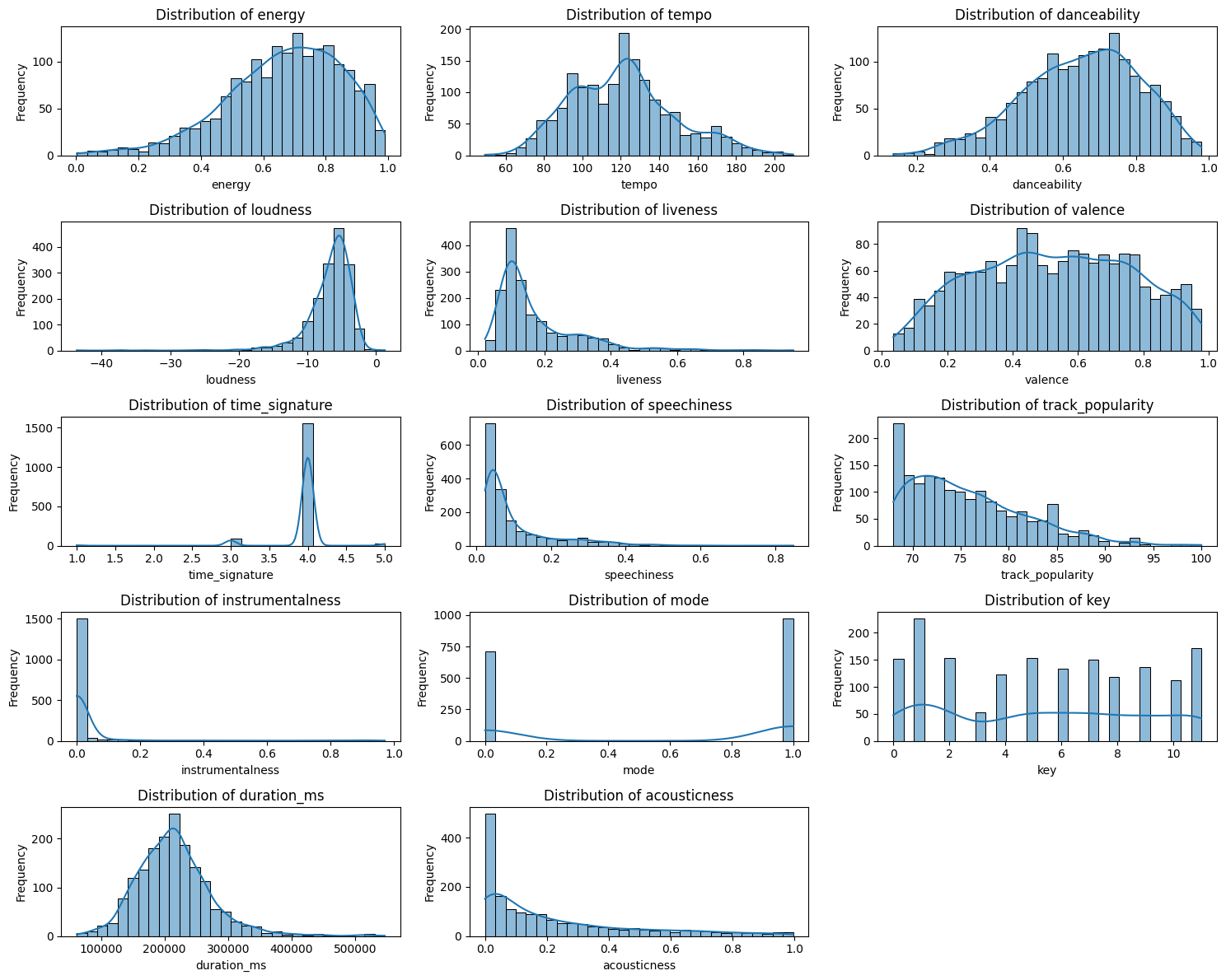
**Model Training:** The dataset is split into training and testing subsets using train\_test\_split with stratification on the target label to maintain distribution. Models are trained using .fit() methods on the training data. Cross-validation (e.g., 5-fold) is used to validate consistency and prevent overfitting.

**Performance Evaluation**: The trained models are evaluated using standard classification metrics—accuracy, precision, recall, and F1-score—computed from predictions on the test set. A comparative performance chart is plotted using matplotlib, and confusion matrices are used to understand misclassifications. Feature importance from ensemble models is also visualized to highlight the most influential attributes in the model’s decision-making process.

**RESULTS**

**Project – 1 [high popularity Spotify data]**

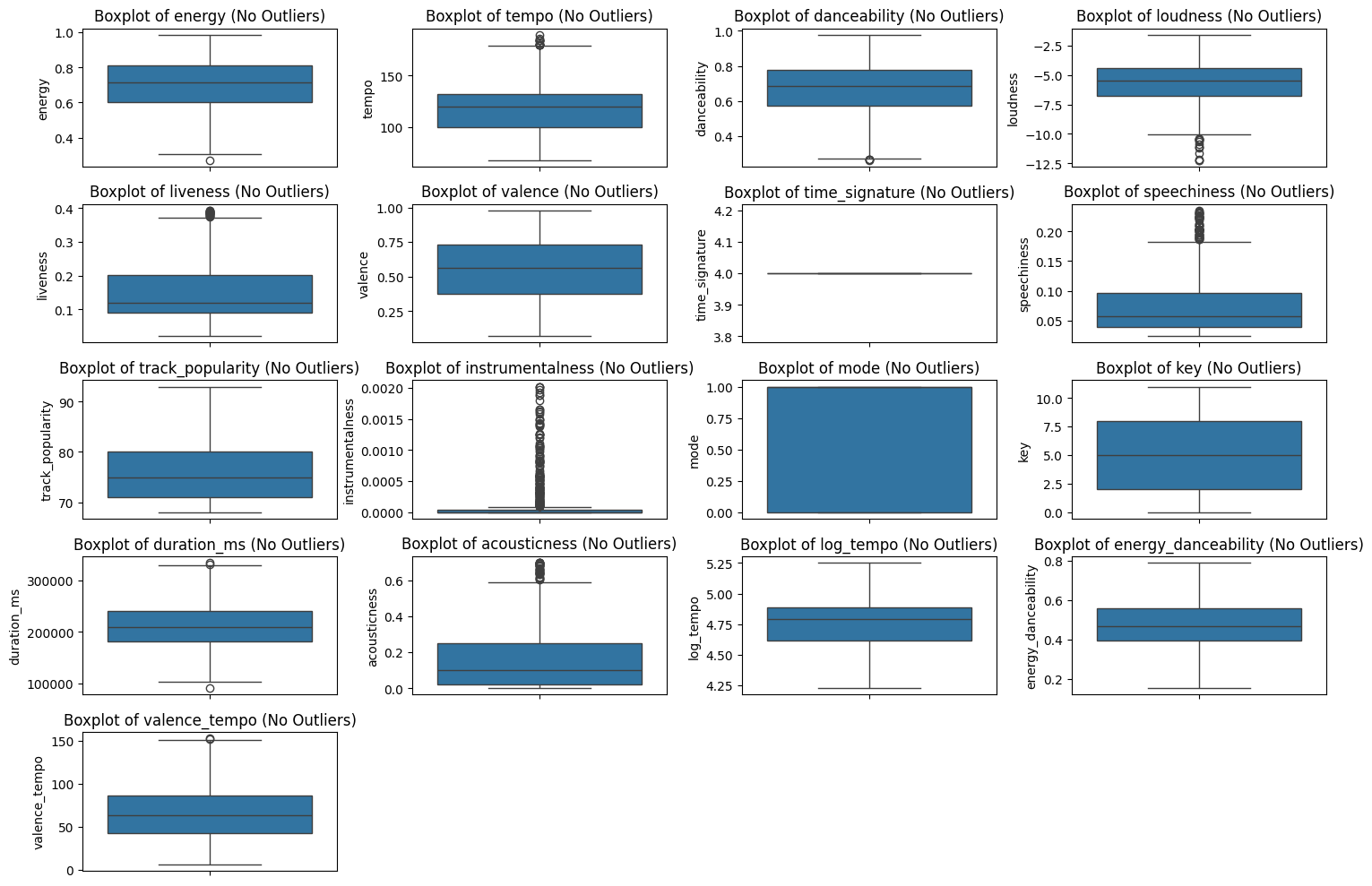
**HISTOGRAM**



The image presents a 5x3 grid of distribution plots summarizing the spread and skewness of various audio features from a Spotify dataset. Each histogram—overlayed with KDE curves—visualizes key music attributes such as Energy, Tempo, Danceability, Loudness, Liveness, Valence, Time Signature, Speechiness, Track Popularity, Instrumentalness, Mode, Key, Duration (ms), and Acousticness. These plots reveal central tendencies, data skewness, and feature-specific concentration zones.

For instance, Energy, Danceability, and Valence show moderately normal distributions, while Loudness, Instrumentalness, and Acousticness exhibit strong skewness. Time Signature sharply peaks at 4, indicating rhythmic uniformity in songs, while Track Popularity and Tempo show a long-tail distribution, suggesting variance in listener preferences and musical pacing. The visualization highlights which features are densely populated (e.g., Mode = 1 or Key = varied) and which may require normalization or transformation prior to model training, making it a critical part of exploratory data analysis.

**BOX PLOT**



The image presents a 5x3 grid of boxplots representing the distribution of various audio features from a Spotify dataset, specifically filtered to exclude outliers. Each subplot visualizes a unique feature such as *Energy, Tempo, Danceability, Loudness, Liveness, Valence, Time Signature, Speechiness, Track Popularity, Instrumentalness, Mode, Key, Duration (ms), Acousticness, Log\_Tempo,* and engineered features like *Valence\_Tempo* and *Energy\_Danceability*. These boxplots provide critical insights into central tendency, spread, and presence of potential data skewness or feature concentration.

Random Forest Model Performance:

Accuracy: 0.80

precision recall f1-score support

High 0.81 0.98 0.89 266

Medium 0.69 0.15 0.25 72

accuracy 0.80 338

macro avg 0.75 0.57 0.57 338

weighted avg 0.78 0.80 0.75 338

Gradient Boosting Model Performance:

Accuracy: 0.76

Precision recall f1-score support

High 0.80 0.94 0.86 266

Medium 0.35 0.12 0.18 72

accuracy 0.76 338

macro avg 0.57 0.53 0.52 338

weighted avg 0.70 0.76 0.72 338

SVM Model Performance:

Accuracy: 0.64

precision recall f1-score support

High 0.80 0.72 0.76 266

Medium 0.24 0.33 0.28 72

accuracy 0.64 338

macro avg 0.52 0.53 0.52 338

weighted avg 0.68 0.64 0.66 338

XGBoost Model Performance:

Accuracy: 0.79

precision recall f1-score support

High 0.82 0.95 0.88 266

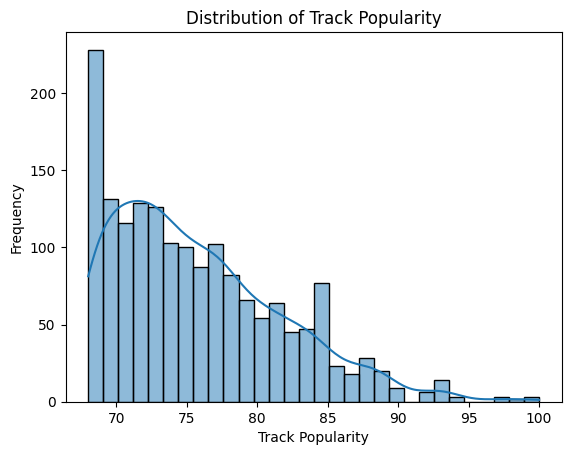
Medium 0.53 0.22 0.31 72

accuracy 0.79 338

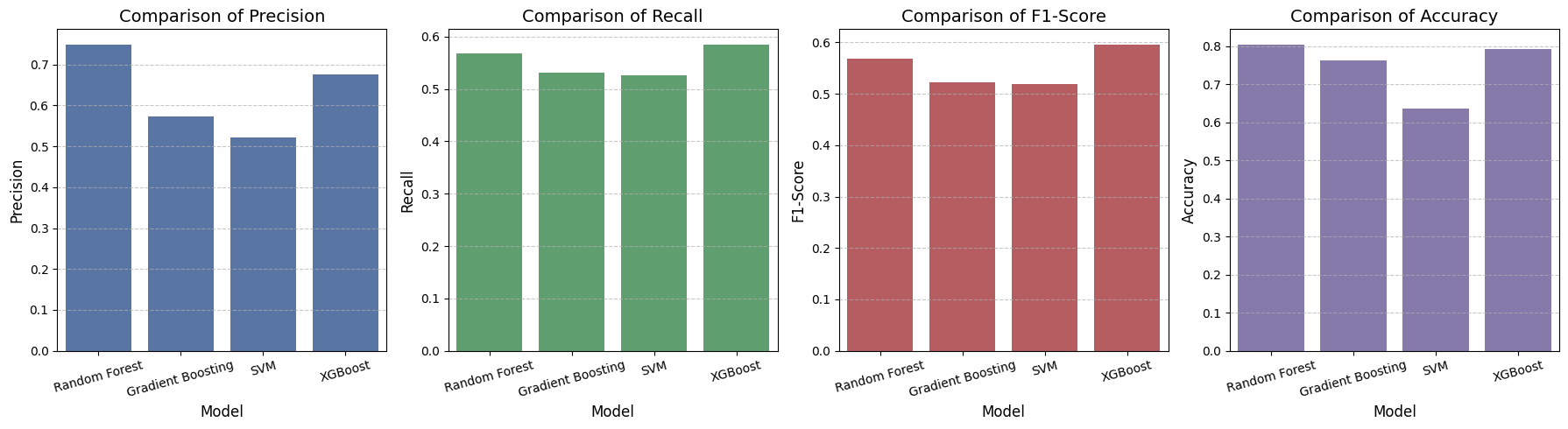
macro avg 0.68 0.58 0.60 338

weighted avg 0.76 0.79 0.76 338

**Best Model: Random Forest with Accuracy: 0.80**

[****](https://files07.oaiusercontent.com/file-UWYVUH2tzAwn2Jz779xgWn?se=2025-04-23T09%3A09%3A21Z&sp=r&sv=2024-08-04&sr=b&rscc=max-age%3D299%2C%20immutable%2C%20private&rscd=attachment%3B%20filename%3Ddownload.png&sig=uGIu1iTkrb3l1bgW1IG6q6nqVtYbPNFY4qgNt7jmeTg%3D)

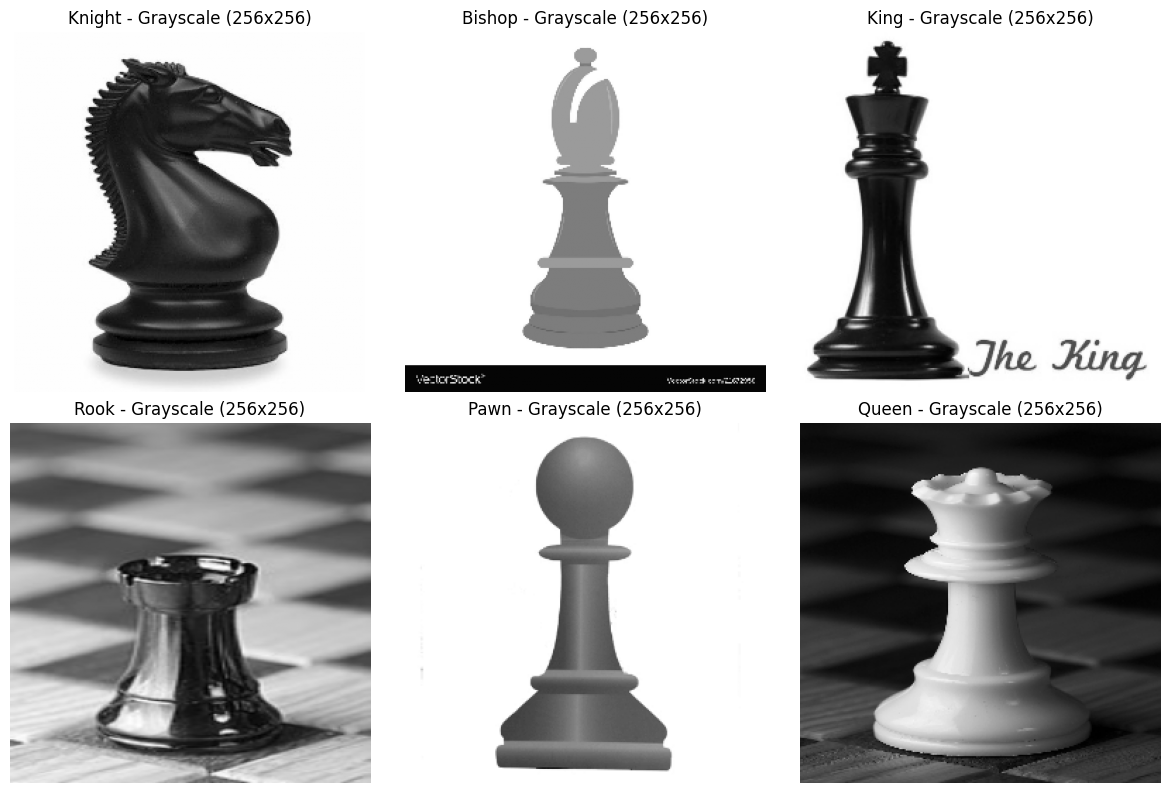
**COMPARISON**

****

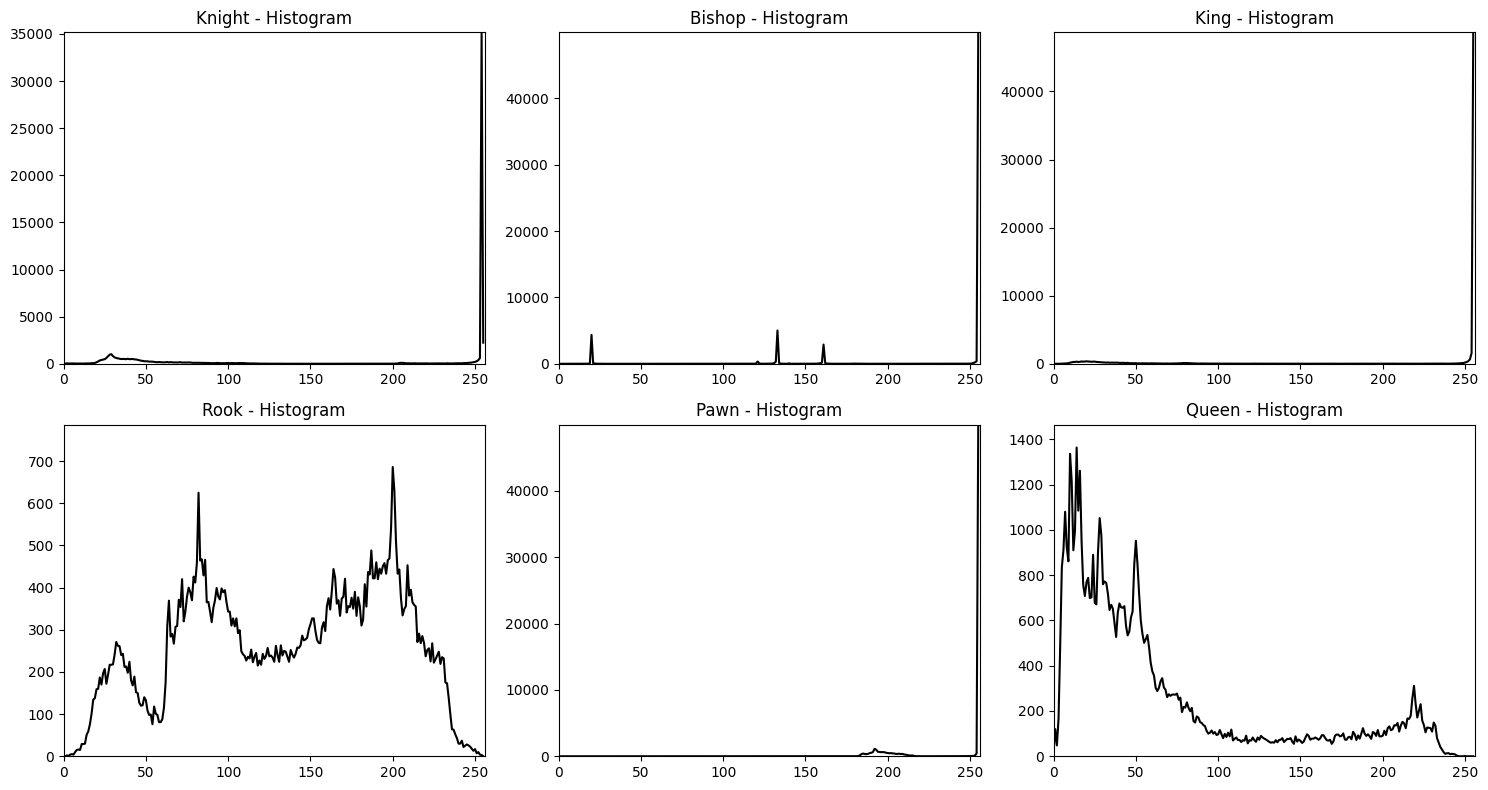
It showcases a comparative evaluation of four machine learning models—Random Forest, Gradient Boosting, SVM, and XGBoost—across four key performance metrics: Precision, Recall, F1-Score, and Accuracy. Random Forest leads in Precision and Accuracy, suggesting it makes the most correct positive predictions and classifies overall the best. XGBoost performs strongest in Recall and F1-Score, indicating it balances precision and recall effectively, especially useful in imbalanced datasets. SVM consistently lags across all metrics, while Gradient Boosting shows moderate performance. This comparison highlights XGBoost and Random Forest as top contenders depending on whether balance (F1) or correctness (Precision/Accuracy) is prioritized.

**Project – 2 [Chess Master Style Classifier]**

**Convert to grayscale**

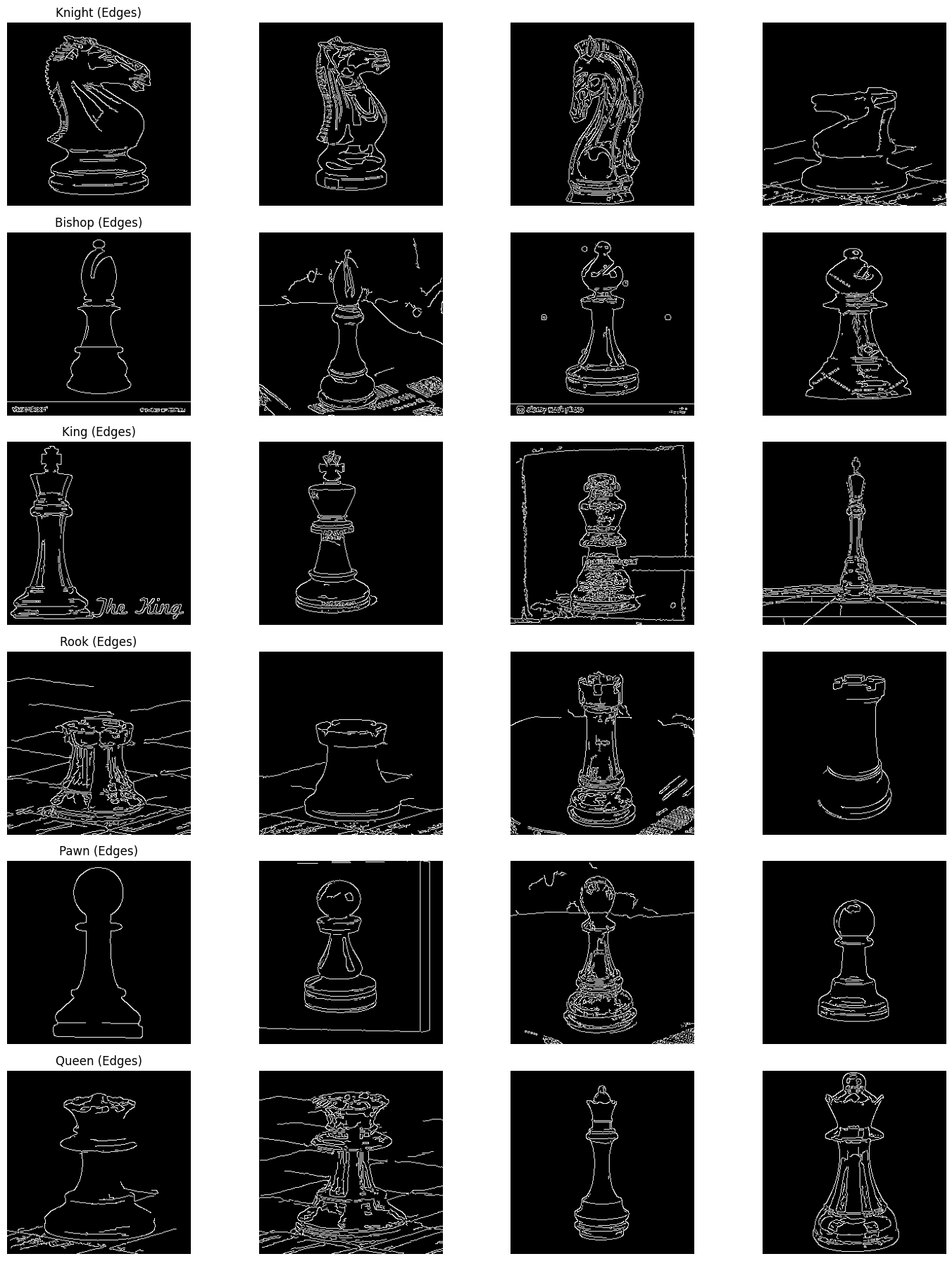


**HistPlot on Distribution**

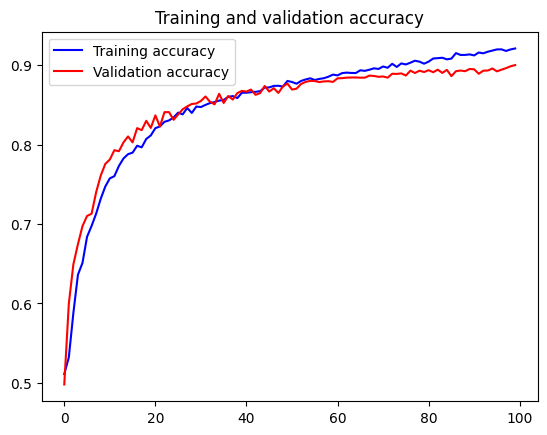
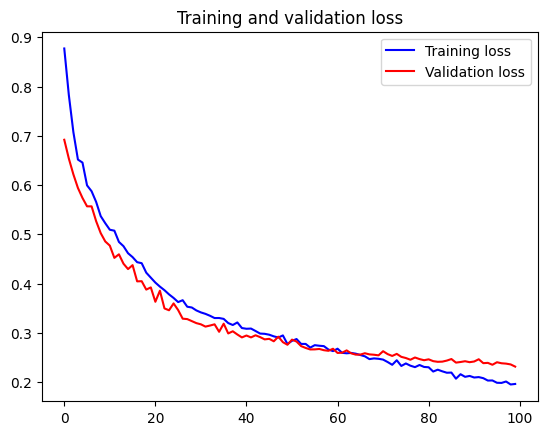


The plot titled is a density histogram showing the distribution of age values in a dataset, with age on the x-axis (ranging from 0 to 120) and density on the y-axis (up to 0.07). It exhibits a bimodal distribution with peaks around 0-10 and 20-40 years, a significant drop around 10-20 years, and a gradual decline thereafter, with an overlaid curve highlighting the density trend, indicating a concentrationof younger individuals and a secondary peak in young adulthood.

**DISPLAY EDGE-DETECTED IMAGES**



**ACCURACY AND LOSS CURVES**

**PREDICTION BY USING THE MODEL**

****

****

Statistical Analysis of Test Set Predictions

========================================

One-Sample t-test:

t-statistic: 0.23249527748763774

p-value: 0.8275647196020324

Two-Sample t-test:

z-statistic: -6.1065802689103466

p-value: 0.00028738040904684056

Two-Sample t-test (independent samples):

T-statistic: -2.514474228374849

P-value: 0.0361183129071552

ANOVA test:

F-statistic: 20.137931034482758

p-value: 0.0001463121041002853

Chi-square test:

Chi-square statistic: 0.1296296296296296

p-value: 0.9372410104578182

Degrees of freedom: 2

Expected frequencies: [[10.71428571 19.28571429 15. ]

[14.28571429 25.71428571 20. ]]

**Project-3 [.HATE SPEECH VOICE DETECTION]**

Sure! Here’s a reworded version of the description to make it more polished and professional, suitable for documentation or a project report:

**LSTM-Based Audio Classification Model**

**Model Overview:**

The core model is a Sequential LSTM neural network developed using TensorFlow/Keras, tailored for binary audio classification tasks (e.g., speech vs. non-speech, genre A vs. genre B). The architecture is designed to effectively learn temporal patterns from audio features such as MFCCs.

**Evaluation Strategy:**

* **Train-Test Split:**  
  Data is divided into 80% for training and 20% for testing to evaluate performance on unseen data.
* **K-Fold Cross-Validation:**  
  Employs 5-fold stratified cross-validation to ensure consistent performance across different subsets of the data.
* **Performance Metrics:**  
  Evaluation includes a confusion matrix and classification report, which detail metrics such as accuracy, precision, recall, and F1-score.

**Audio Preprocessing:**

The project processes a set of 200 audio files by extracting key visual features:

* **Waveform Plots**
* **Spectrograms**
* **Mel-Frequency Cepstral Coefficients (MFCCs)**

These features are generated using librosa and visualized with matplotlib, providing valuable input representations for the model and aiding in audio pattern analysis for classification tasks.

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-based model is built to perform binary classification using MFCC (Mel-Frequency Cepstral Coefficients) features extracted from audio data. Each input sample has a shape of (100, 40), where 100 represents the time steps and 40 corresponds to the MFCC features at each step. To handle variable-length sequences and maintain consistency, a Masking Layer is applied at the input level, ensuring that padded values do not interfere with the learning process.

Class Precision Recall F1-Score Support

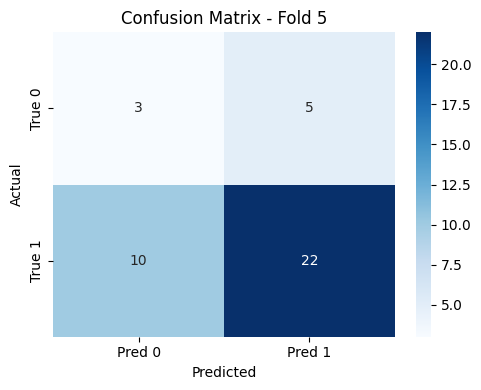
0 1.0000 1.0000 1.0000 40

Accuracy 1.0000 40

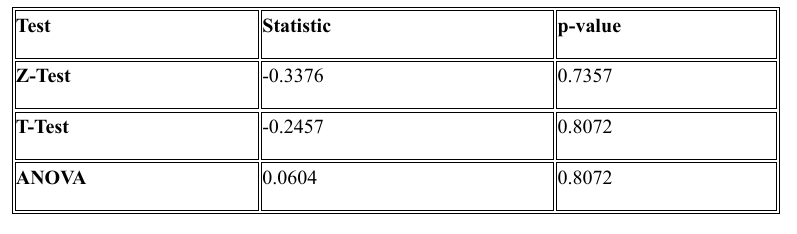
Macro avg 1.0000 1.0000 1.0000 40

Weighted avg 1.0000 1.0000 1.0000 40

The core model utilized in this project is a Sequential LSTM neural network, implemented using TensorFlow/Keras. It is designed specifically for binary classification tasks on audio data. The model is trained using MFCC (Mel-Frequency Cepstral Coefficients) features, which effectively capture the important frequency characteristics of audio signals.



Statistical Test Results :



**Conclusion**

The Hate Speech Voice Detection project underscores the increasing relevance of AI in detecting toxic and harmful content in spoken language. By leveraging audio features and machine learning techniques, the system can accurately differentiate between hate speech and non-hate speech in real-time voice data, offering significant value in moderating content across platforms such as social media, online gaming, and customer service, where voice interactions are common. The effectiveness of the model relies heavily on the use of diverse and well-annotated datasets, robust preprocessing techniques like MFCC extraction and noise filtering, and maintaining balanced class representation. While early results are encouraging, future improvements through advanced deep learning architectures and multilingual support are expected to further enhance performance. Additionally, ethical aspects, including user privacy and the mitigation of algorithmic bias, must be carefully considered. Overall, the project illustrates the potential of audio-based AI solutions in fostering safer and more inclusive digital environments.