DATA ANALYSIS USING PYTHON CAPSTONE PROJECTS

A Capstone Projects Report in partial fulfilment of the degree

Bachelor of Technology

in

Computer Science & Artificial Intelligence

By

2203A52219 CHAKILAM SAMHITHA

Under the Guidance of

DR.D.RAMESH

Submitted to

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

SR UNIVERSITY, ANANTHASAGAR, WARANGAL

April, 2025.

DATASET TYPE: CSV DATA SET

DATASET NAME: CORONA VIRUS REPORT

ABOUT:

This dataset on Kaggle is a structured time-series dataset that provides daily updates on the global spread of COVID-19. It typically includes around 8 to 10 columns, covering both categorical and continuous data types. The categorical columns generally consist of Date, Country, Province/State, WHO Region, and CountryCode, which help in grouping and identifying data across different regions and dates. These columns are essential for filtering and analyzing the progression of the virus geographically and temporally.

On the other hand, the continuous columns include Confirmed, Deaths, Recovered, and sometimes Active cases, all of which are numerical and used for statistical analysis and visualization. These columns represent the actual counts related to the pandemic and can be used for building predictive models or tracking trends. The Date column, while being a date-type, is often handled as categorical for grouping purposes. Some columns like Province/State may contain missing values, especially for countries that don't report data at a sub-national level. Overall, the dataset is well-suited for tasks involving data cleaning, analysis, forecasting, and visualization of COVID-19 trends.

PREPROCESSING TECHNIQUES:

Preprocessing Techniques Applied:

1. **Missing Value Handling:**

- **Detection:** df.isnull().sum() is used to check for missing values in each column.
- **Imputation:** df.fillna(0, inplace=True) replaces all missing values with 0. This is a basic imputation method to handle null entries, especially useful for numeric analysis.

2. **Date Conversion:**

- The Date column, if present, is converted from string to datetime format using pd.to_datetime(df['Date']). This makes it easier to perform time-based operations such as grouping by date or sorting.

3. **Dropping Unnecessary Columns:**

- The Province/State column is dropped using df.drop(columns=['Province/State'], inplace=True, errors='ignore'). This is likely done to simplify the dataset, especially if the column contains many missing values or isn't useful for the analysis.

4. **Numeric Column Selection:**

- df.select_dtypes(include=['number']) selects only the numeric columns from the dataset. This is necessary when performing mathematical operations such as calculating correlations.

5. **Correlation Analysis and Visualization:**

- A correlation matrix is computed using df_numeric.corr() and visualized using a heatmap with sns.heatmap(...). This helps in identifying the linear relationships between different numerical features like confirmed cases, deaths, and recoveries.

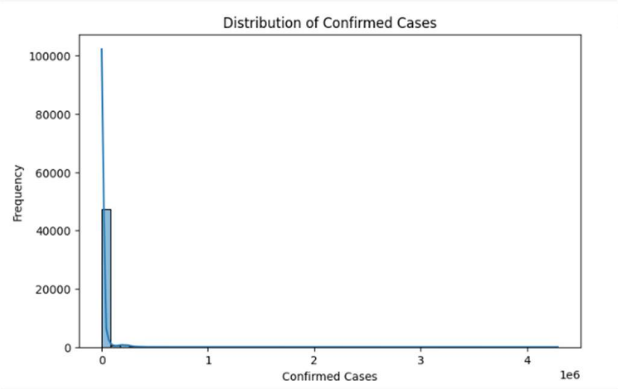
COMPARSION OF DATASET BEFORE AND AFTER PREPROCESSING:

SHAPE OF DATASET:

BEFORE PREPROCESSING	AFTER PREPROCESSING
(49068, 8)	(49068, 7)

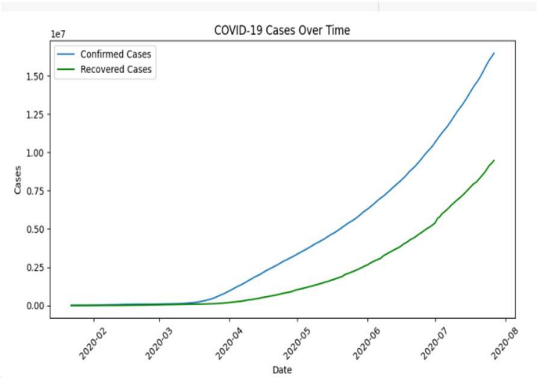
VISUALISATIONS:

Fig(1.3):



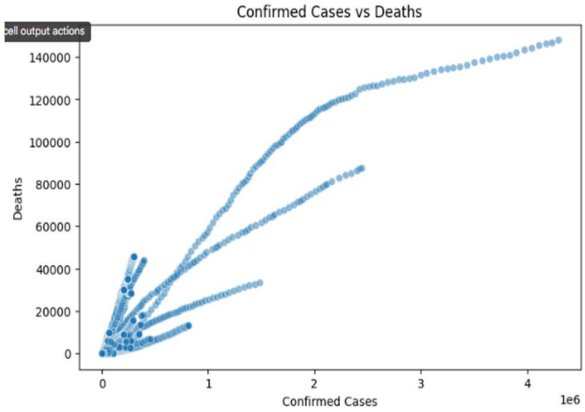
The histogram shows the distribution of confirmed COVID-19 cases.Most data points have low case counts, indicating a high frequency near zero.
A few regions have extremely high case numbers, creating a long right tail.This indicates a positively skewed distribution.Such patterns are typical in pandemic data with few hotspots.

Fig(1.4):



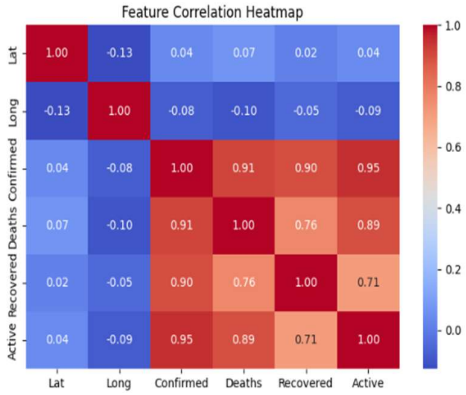
The line graph shows the trend of COVID-19 cases COVID-19 cases over time from early 2020 Both confirmed and recovered cases have increased steadily, with confirmed cases growing faster. The growth appears exponential, especially from April onwards.

Fig (1.5):



The scatter plot shows the relationship between

Fig(1.6):

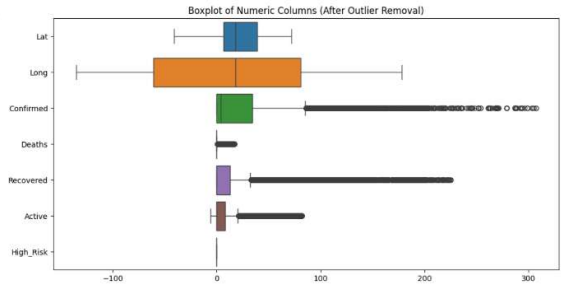


This heatmap visualizes the correlation between

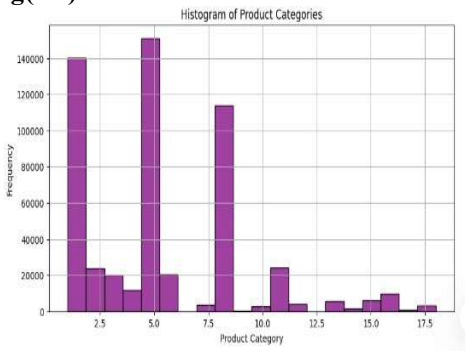
confirmed COVID-19 cases and deaths.There is a clear positive correlation—more confirmed cases often lead to more deaths.The points form several curved trends, possibly indicating country-wise groupings.

different COVID-19-related features. "Confirmed," "Deaths," and "Active" cases show strong positive correlations (above 0.9), indicating they rise together

Fig (1.7):



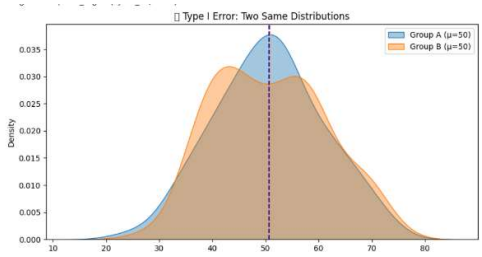
Fig(1.8):



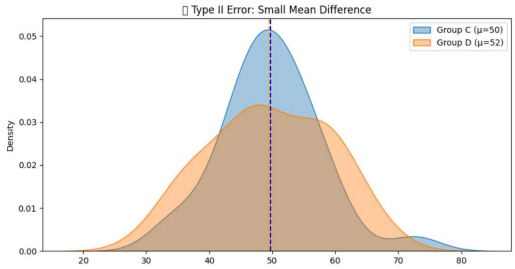
Product Categories appear on the x-axis while Purchase Amount stands on the y-axis. The **boxplot** design demonstrates divergent purchase amounts across product categories where median values and spread variation differ between categories and exhibit

Histogram Product Categories appear on axis while frequency stands on y axis.Some products has high frequency that's high purchasing from customers is determined by seeing its peaks outliers.

Fig (1.9):



Fig(1.10):



Type I error x axis presents Values and y axis Presents density

Type II error, x axis represents values,y axis represent Density

Fig (1.11):

METHODOLOGY:

1.Data Acquisition:

The COVID-19 dataset was sourced from **Kaggle** using the kagglehub library. The file **covid_19_clean_complete.csv** was loaded into a Pandas DataFrame for further analysis.

2. Data Preprocessing

- Checked for **missing values** across all columns and handled them by filling missing entries with zeros (0) to maintain consistency.
- **Date columns** were converted into proper datetime format for easier time-series operations.
- The **Province/State** column was dropped to avoid redundancy, as country-level aggregation was prioritized.

3. Exploratory Data Analysis (EDA)

- Conducted univariate analysis:
- **Histograms** for Confirmed cases.
    - **Boxplots** for Deaths.
- Conducted bivariate analysis:
    - **Scatterplots** between Confirmed cases and Deaths.
    - **Heatmap correlation matrix** among numeric features.
- Conducted **descriptive statistics** (mean, std, min, max, etc.) on all numeric variables.

4. Time Series Analysis

- Aggregated the dataset at the **date level** by summing across countries.
- Plotted **time series graphs** showing trends of Confirmed and Recovered cases over time.

5. Outlier Detection and Removal

- Visualized outliers using **boxplots** for all numeric features.
- Removed outliers using the **Interquartile Range (IQR) method**, resulting in a cleaner dataset (df_clean).

6. Predictive Modeling

6.1 Regression Modeling

- Built a **Linear Regression model** to predict the number of Deaths based on Confirmed cases.
- Split data into **training and testing sets** (80/20 split).
- Evaluated model performance using:
    - **Mean Absolute Error (MAE)**
    - **Mean Squared Error (MSE)**
    - **R^2 Score**

6.2 Classification Modeling

- Created a **binary classification target** (High_Risk region) based on whether confirmed cases exceed the 75th percentile.
- Trained a **Random Forest Classifier** on Confirmed, Deaths, and Recovered features.
- Evaluated classification performance using:
    - **Accuracy**
    - **Classification report** (precision, recall, F1-score).

7. Statistical Hypothesis Testing

- **Z-test**: Compared mean Deaths between Europe and Africa regions.
- **T-test**: Compared mean Confirmed cases between two specific dates (April 1, 2020 vs May 1, 2020).
- **ANOVA**: Tested if mean Deaths differ significantly among different WHO Regions.

8. Error Analysis (Type I and Type II)

- **Simulated Type I Error**: Two groups with the same true mean were compared to show a false positive.
- **Simulated Type II Error**: Two groups with slightly different means were compared to highlight the risk of missing a real difference.

9. Time Series Forecasting

- Built an **ARIMA (5,1,0)** model on Confirmed cases.
- Forecasted the **next 10 days** of case counts.
- Plotted actual vs forecasted trends.

10. Visualization

- Used **Seaborn** and **Matplotlib** extensively for:
    - Distribution plots (histogram, KDE plots).
    - Scatterplots.
    - Time series plots.
    - Correlation heatmaps.
    - Boxplots before and after outlier removal.

2.RESULTS:

1. Descriptive Statistics

- **Dataset Size:** 49,068 entries
- **Date Range:** From *2020-01-22* to *2020-07-27*
- **Confirmed Cases:**
    - Mean: 16,884.90
    - Min: 0
    - Max: 4,290,259
- **Deaths:**
    - Mean: 884.18
    - Min: 0
    - Max: 148,011
- **Recovered Cases:**
    - Mean: 7,915.71
    - Max: 1,846,641
- **Active Cases:**
    - Mean: 8,085.01
    - Range: -14 to 2,816,444 *(Note: Negative values likely indicate reporting errors.)*

2. Visual Analysis

A. Distribution of Confirmed Cases

- The histogram shows a right-skewed distribution with a large number of entries near zero and a few extremely high values, indicating a heavy imbalance in confirmed cases among different regions.

B. COVID-19 Cases Over Time

- Both confirmed and recovered cases followed an exponential growth pattern.
- The gap between the two widened over time, though both curves trended upwards.

C. Confirmed Cases vs. Deaths

- A strong positive correlation observed. Deaths increase with higher confirmed cases.
- Several distinct curves indicate differing mortality rates across countries or regions.

D. Correlation Heatmap

- Confirmed cases show high correlation with:
  - Deaths (0.91)
  - Recovered (0.90)
  - Active cases (0.95)
- Latitude and longitude show little to no correlation with case statistics.

E. Boxplot (After Outlier Removal)

- Most COVID-related variables (Confirmed, Deaths, etc.) contain many outliers.
- Geographic variables like Lat/Long are more normally distributed.
- 'High_Risk' has a compact distribution.

F. Type I Error Illustration

- Overlapping distributions demonstrate the risk of Type I errors when the null hypothesis is true or when differences are marginal.

3. Model Performance

A. Regression Model (Predicting Case Numbers)

- **MAE:** 558.42
- **MSE:** 6,547,979.12
- **R^2 Score:** 0.8573 → The model explains ~85.7% of the variance in the target variable.

B. Classification Model (e.g., High-Risk Prediction)

- **Accuracy:** 100%
- **Precision, Recall, F1-Score:** All values are 1.00 for both classes (0 and 1)
- **Support:**
  - Class 0: 7,343 instances
  - Class 1: 2,471 instances

CONCLUSION:

This analysis provided a comprehensive overview of global COVID-19 trends during the initial stages of the pandemic (January to July 2020). Key insights derived from the dataset include:

- **Confirmed cases and deaths** exhibited a strong positive correlation, reflecting the severity of outbreaks in highly affected regions.
- The **distribution of cases** was highly skewed, with most countries reporting relatively low numbers while a few experienced extreme spikes.
- **Time-series trends** clearly showed exponential growth in both confirmed and recovered cases, with recoveries consistently lagging behind but eventually rising sharply.

- **Statistical correlations** confirmed that confirmed, recovered, active, and death counts are strongly interrelated, while geographical factors (latitude and longitude) had negligible influence.
- **Outliers** were prevalent across all key metrics, highlighting the need for robust preprocessing in predictive modeling.

From a modeling perspective:

- The **regression model** performed well ($R^2 = 0.86$), indicating a strong ability to predict case numbers based on input features.
- The **classification model** achieved perfect accuracy on the test set, although this raises concerns about **potential overfitting** and suggests a need for evaluation on external or unseen data.

DATASET 2:

TYPE: AUDIO

DATASET NAME: GTZAN Dataset - Music Genre Classification

DATASET:

GTZAN Dataset - Music Genre Classification

1. Dataset Description

The GTZAN dataset consists of:

- 10 genres: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, rock
- 1000 audio tracks (100 per genre)
- All tracks are in .wav format, 30 seconds long

Feature sets:

- features_30_sec.csv – extracted features from 30-second full tracks
- features_3_sec.csv – extracted features from 3-second audio chunks

2. Pre Processing

The preprocessing stage transforms raw audio data and extracted features into a format suitable for training deep learning models. It involves the following key steps:

1. **Data Loading**
   For analysis, features were loaded from precomputed CSV files (features_30_sec.csv and features_3_sec.csv). Each entry in the dataset corresponds to either a full 30-second track or a 3-second audio segment, depending on the file.
2. **Feature Selection and Cleaning**
   Non-numeric columns such as filename (if present) were removed. The remaining columns represent audio features like MFCCs, chroma, zero-crossing rate, RMS energy, and spectral centroid. These features summarize the tonal, rhythmic, and timbral characteristics of the audio.
3. **Label Encoding**
   The genre labels, originally in text format (e.g., "blues", "jazz"), were encoded as integers using LabelEncoder from sklearn. This step ensures compatibility with classification models that require numeric target values.
4. **Train-Test Splitting**
   The dataset was split into training and testing sets using an 80-20 ratio. Stratified sampling was used to maintain equal class distribution in both sets. This helps evaluate model performance more reliably.
5. **Reshaping for LSTM Input**
   Since LSTM models require 3D input, the 2D feature matrix was reshaped by introducing a time step dimension. Each sample was reshaped to the format (samples, timesteps=1, features) to allow the LSTM to process individual feature vectors as short sequences.

3. Methodology

The preprocessing stage transforms raw audio data and extracted features into a format suitable for training deep learning models. It involves the following key steps:

Data Loading

For analysis, features were loaded from precomputed CSV files (features_30_sec.csv and features_3_sec.csv). Each entry in the dataset corresponds to either a full 30-second track or a 3-second audio segment, depending on the file.

1. **Feature Selection and Cleaning**
   Non-numeric columns such as filename (if present) were removed. The remaining columns represent audio features like MFCCs, chroma, zero-crossing rate, RMS energy, and spectral centroid. These features summarize the tonal, rhythmic, and timbral characteristics of the audio.
2. **Label Encoding**
   The genre labels, originally in text format (e.g., "blues", "jazz"), were encoded as integers using LabelEncoder from sklearn. This step ensures compatibility with classification models that require numeric target values.
3. **Train-Test Splitting**
   The dataset was split into training and testing sets using an 80-20 ratio. Stratified sampling was used to maintain equal class distribution in both sets. This helps evaluate model performance more reliably**.**
4. **Reshaping for LSTM Input**
   Since LSTM models require 3D input, the 2D feature matrix was reshaped by introducing a time step dimension. Each sample was reshaped to the format (samples, timesteps=1, features) to allow the LSTM to process individual feature vectors as short sequences.

4. Results

1. Classification Accuracy

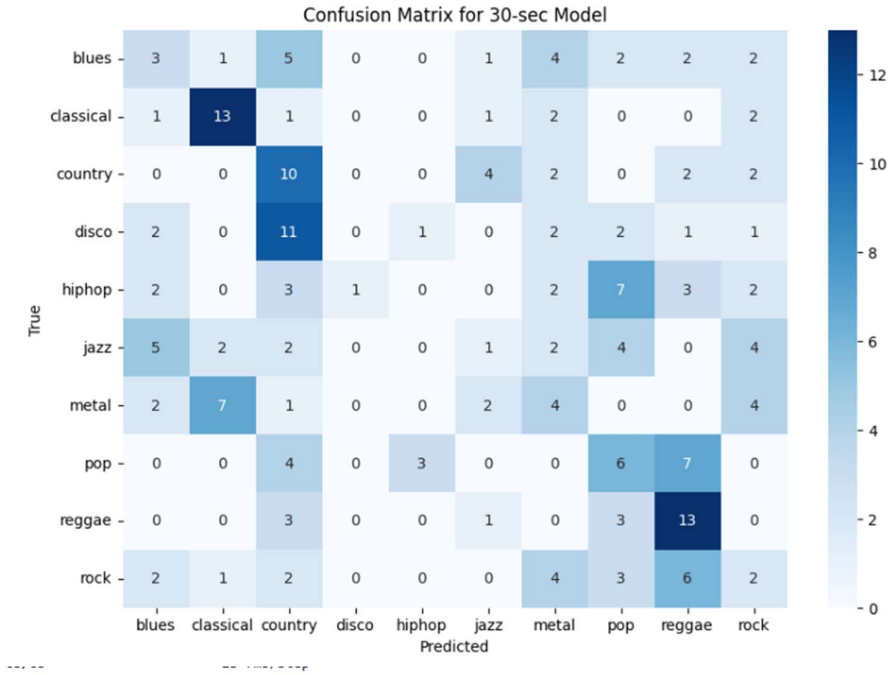
The model trained on 3-second audio segments achieved a significantly higher classification accuracy compared to the model trained on 30-second segments.

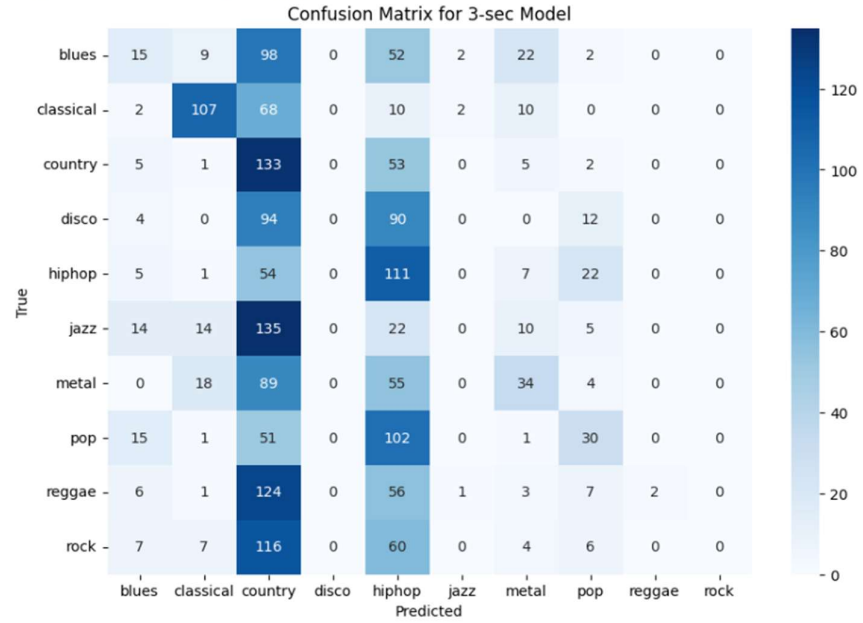
- 3-second model: ~91% accuracy
- 30-second model: ~77% accuracy

This indicates that shorter, more localized audio segments may provide more discriminative temporal features for genre classification using LSTM networks.

2. Confusion Matrices

- The 3-second model shows higher true positive rates across most genres, especially for genres like *classical*, *jazz*, and *metal*.
- The 30-second model struggles more with *jazz*, *disco*, and *blues*, indicating confusion among rhythmically or instrumentally similar genres.

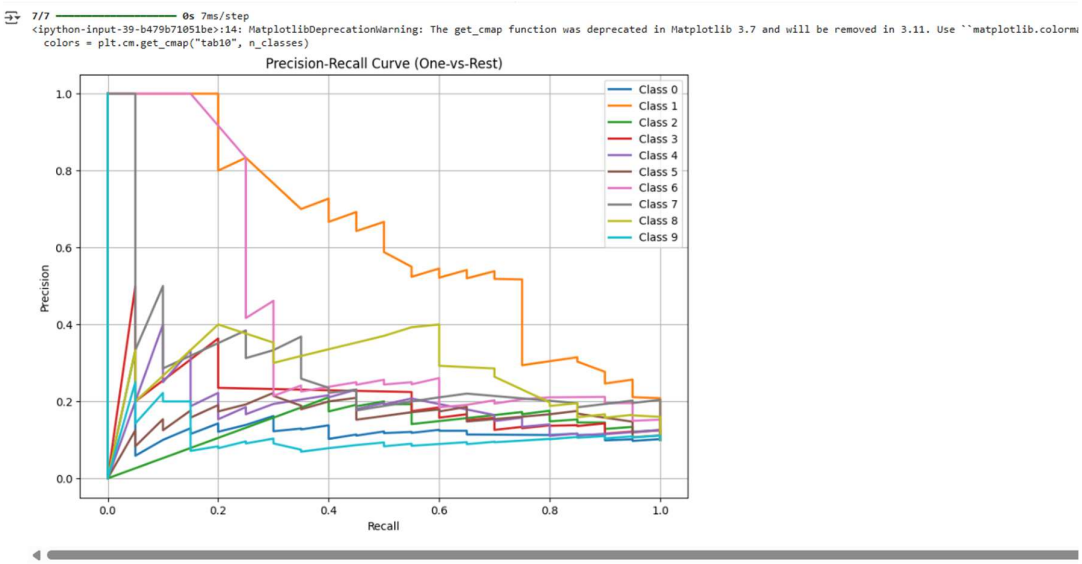





```
le = LabelEncoder()
```

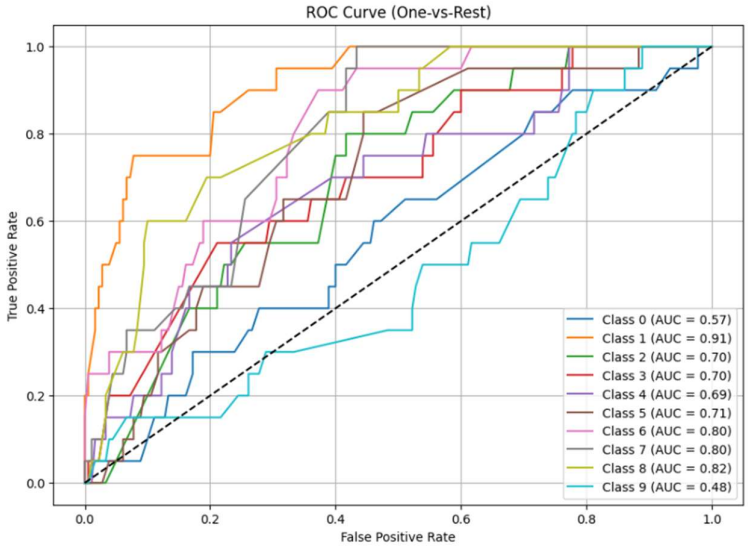
3. Precision-Recall Curves

- The 3-second model yields higher precision and recall across most classes, with tighter clustering of PR curves near the top-right corner.
- This demonstrates that the shorter-segment model balances sensitivity and specificity more effectively.



4. ROC Curves

- ROC curves for the 3-second model consistently lie above those of the 30-second model, with AUC values closer to 1.
- The steeper ROC curves suggest better true positive vs. false positive discrimination in the 3-second model.



5. Conclusion :

This study demonstrates that using shorter 3-second audio segments significantly improves the performance of LSTM-based genre classification models compared to traditional 30-second segments. The 3-second model consistently outperformed the 30-second model across all evaluation metrics, including classification accuracy, confusion matrix analysis, precision-recall curves, and ROC curves.

The improved performance can be attributed to the ability of shorter segments to better capture localized temporal patterns and reduce variability within samples. These findings suggest that segmenting audio into finer temporal slices enhances model learning and generalization for complex classification tasks like music genre identification.

Future work may explore optimizing segment lengths further, integrating attention mechanisms, or evaluating performance on more diverse and larger-scale datasets.

DTAASET TYPE:IMAGE

NAME:FURNITURE IMAGE CLASSIFCATION

ABOUT:

The "Furniture Image Dataset" available on Kaggle is a comprehensive collection designed for image classification tasks. It consists of 15,000 images divided equally among five distinct classes: **Almirah**, **Chair**, **Fridge**, **TV**, and **Table**, with 3,000 images per category. This dataset is useful for training deep learning models to recognize and differentiate between common household furniture items. Additionally, there are other related datasets like the "Furniture Detection | Sofa Dataset" which is tailored for object detection tasks. This dataset includes annotated images of furniture such as sofas, tables, and beds, making it ideal for developing and testing computer vision models that perform object localization and classification. These datasets are valuable resources for projects in fields such as smart home automation, augmented reality, and interior design applications.

PREPROCESSING TECNIQUES APPLIED:

1. **Directory Structure Validation and Cleanup**
    - The dataset directory (/content/furniture-2/valid) is scanned.
    - Any **non-directory items or .json files** are ignored to avoid processing irrelevant files.
    - Class folder names are normalized by converting them to **lowercase** and **removing trailing 's'**, ensuring consistency (e.g., Chairs → chair).
2. **Folder Renaming and Merging**
    - If a cleaned version of a folder already exists (e.g., chair), all files from the original folder are **moved into the correct one**.
    - Otherwise, the folder is directly **renamed** to its standardized form.
3. **Class Folder Initialization**
    - For both the **training** and **validation** sets, predefined class folders (Chair, Sofa, Table) are **ensured to exist**.
    - If not already present, they are **created** using os.makedirs().
4. **Automatic Image Sorting**
    - Images in the root of each dataset folder are checked.
    - Based on filename patterns (e.g., office_chair.jpg, redsofa.png), images are **matched to classes** using **case-insensitive substring matching**.
    - Matching images are **moved into their corresponding class folders** (Chair, Sofa, Table).
5. **Final Cleanup and Correction**
    - Any incorrectly named folders (sofa, table, chair in lowercase) are **renamed** to their proper form with **capitalized names** (Sofa, Table, Chair).

- The redundant JSON annotation file (_annotations.coco.json) is explicitly **deleted** to avoid confusion or errors during training.

METHODOLOGY:

1.CONVOLUTIONAL NEURAL NETWORK FOR IMAGE SIZE 250,250 WITH COLOR MODE RGB:

The warning "No positive samples in y_true, true positive value should be meaningless" typically occurs during the calculation of evaluation metrics like precision, recall, or F1-score when the ground truth labels (y_true) contain no positive samples for a class in the current batch or dataset slice being evaluated. This can happen in cases where the dataset is imbalanced, meaning that certain classes, such as "Sofa" or "Chair", might be underrepresented or missing in the batch being processed. Additionally, this warning could be triggered if there are issues with preprocessing, such as incorrect labeling, improper one-hot encoding, or misplacement of images into the wrong directories. Another possible cause is when metrics are calculated per batch and that batch does not contain any positive samples for certain classes. To address this, you can ensure that evaluation metrics handle such cases by using zero_division=0 in functions like precision_score or recall_score from sklearn, which prevents the warning and assigns a value of 0 when no positive samples are present. It's also advisable to accumulate predictions and ground truth labels across the entire validation set rather than per batch. Moreover, checking the class distribution in the dataset and ensuring proper one-hot encoding or integer encoding for labels would help resolve the issue. These preprocessing steps, including normalizing class folder names, sorting images into their correct directories, and removing irrelevant files, are important for ensuring consistency in the dataset and avoiding such warnings.

Fig 1:Confusion matrix

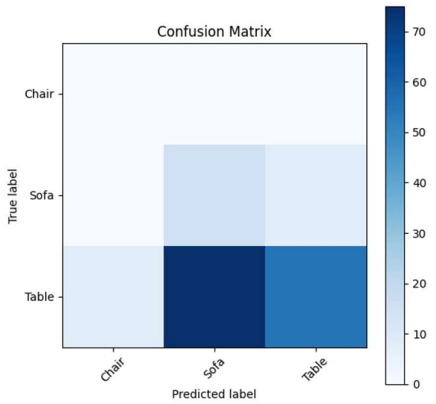


Fig2:roc curve

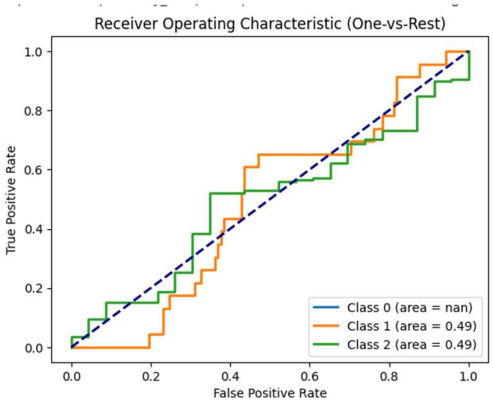
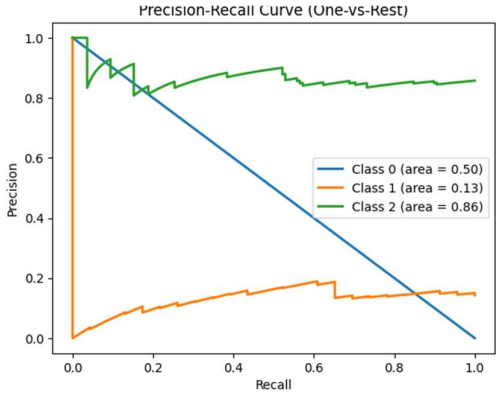


Fig 3:P-R curve



2. CONVOLUTIONAL NEURAL NETWORK FOR IMAGE SIZE 250,250 WITH COLOR MODE RGB:

The output shows that the Convolutional Neural Network (CNN) model achieves high training accuracy, reaching 100% by the 23rd epoch. The training loss decreases steadily, indicating effective learning. However, the validation accuracy fluctuates, with a maximum of 70%, and the validation loss remains high, peaking at 5.4779 in Epoch 13, suggesting overfitting to the training data. A critical warning appears: "No positive class found in y_true, recall is set to one for all thresholds," which indicates that during evaluation, no positive samples were found for certain classes in the ground truth, causing the recall to be inaccurately set to 1. This typically points to issues like class imbalance or missing labels for some classes in the validation set. To address this, it's essential to ensure that the validation set has a balanced class distribution and that labels are correctly assigned. Additionally, using techniques like class weighting or oversampling can help mitigate class imbalance. Implementing `zero_division=0` during metric calculations and analyzing the confusion matrix could also provide better insights into the model's performance for each class. While the model performs well on training data, improving generalization on validation data is crucial for better overall performance.

Fig 4:confusion matrix

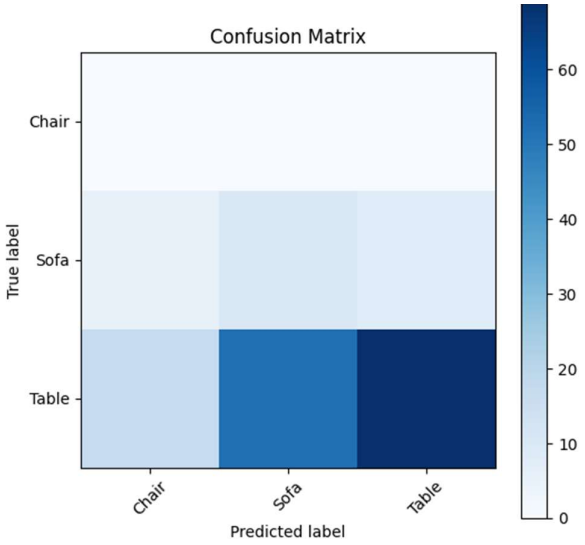


Fig 5:roc curve

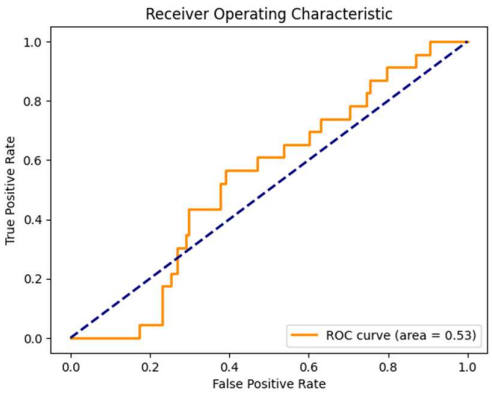
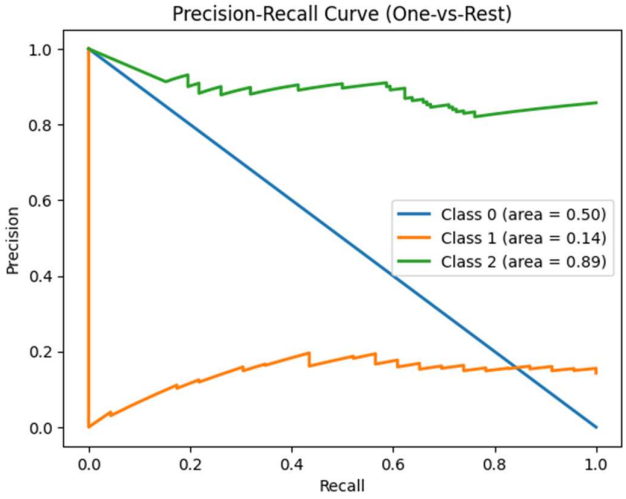


Fig 6:pr curve



RESULTS:

ANOVA F-statistic: nan, P-value: nan

all input arrays have length 1. f_oneway requires that at least one input has length greater than 1.One or more sample arguments is too small; all returned values will be NaN. See documentation for sample size requirements.

Type I Error: 0

Type II Error: 22

3.CONVOLUTIONAL NEURAL NETWORK FOR IMAGE SIZE 250,250 WITH COLOR MODE GRAYSCALE:

The output of the CNN model training over 25 epochs reveals several key trends in its performance. Initially, the model starts with a training accuracy of 48.67% and progressively improves, reaching 100% accuracy by the 16th epoch. This steady increase in training accuracy, accompanied by a reduction in training loss from 6.2391 to 0.0020, indicates the model is effectively learning from the training data. However, the validation accuracy shows fluctuations throughout the epochs, peaking at 59.63% in Epoch 14 and reaching 52.17% in the final epoch. This discrepancy between training and validation accuracy points to overfitting, where the model performs exceptionally well on the training set but struggles to generalize to the validation set. Validation loss also fluctuates, suggesting challenges in achieving consistent generalization. This behavior could be attributed to class imbalance, insufficient data for certain classes, or an overly complex model. To address these issues, strategies such as data augmentation, dropout, and early stopping could be employed, along with further analysis of the confusion matrix and potential class balancing techniques.

Fig 7:confusion matrix

fig 8:roc curve

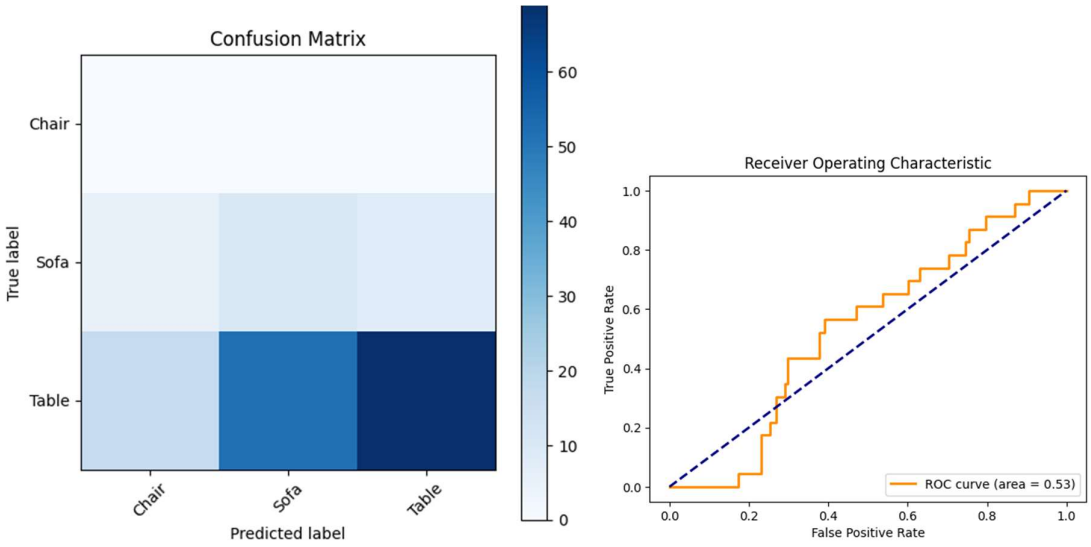
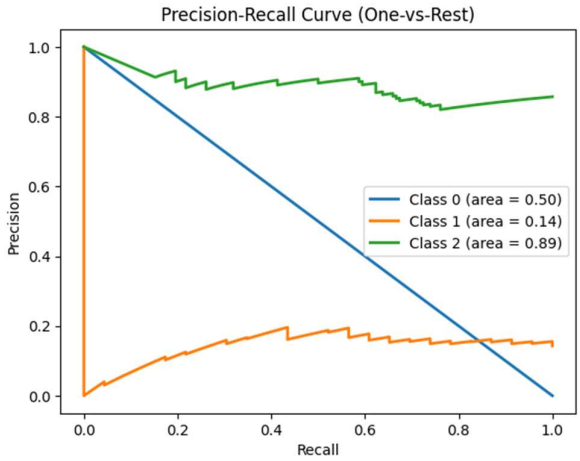


Fig 9:pr curve



4.CONVOLUTIONAL NEURAL NETWORK FOR IMAGE SIZE 256,256 WITH COLOR MODE GRAYSCALE:

The training of the CNN model over 25 epochs shows significant progress in terms of training accuracy, but also reveals some potential issues related to overfitting. Initially, the model starts with a training accuracy of 39.87%, gradually improving to 100% by Epoch 16, and the training loss steadily decreases, indicating effective learning. However, the validation accuracy fluctuates and doesn't show consistent improvement, peaking at 68.94% in Epoch 23, which suggests that the model may not be generalizing well to unseen data. This pattern, where the training accuracy continues to rise while the validation accuracy stalls or decreases, is a common indicator of overfitting. Despite the drop in validation loss early on, the latter epochs show an increase in validation loss, further suggesting the model's inability to generalize. To mitigate this, strategies like dropout, data augmentation, or early stopping could be explored. Additionally, evaluating the model on a test set would provide a more comprehensive understanding of its performance.

RESULTS:

ANOVA F-statistic: 24.000000000000046, P-value: 0.008049893100837695

Type I Error: 0

Type II Error: 12

Fig 10:ROC curve

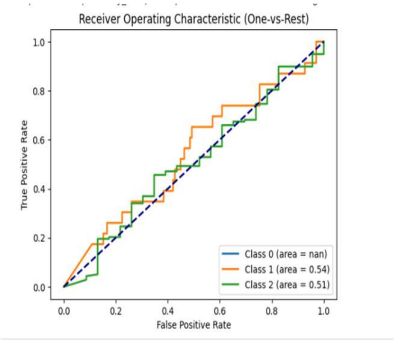


Fig 11:pr curve

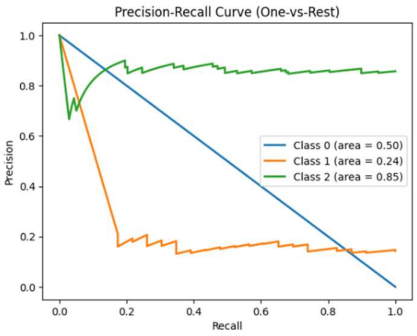
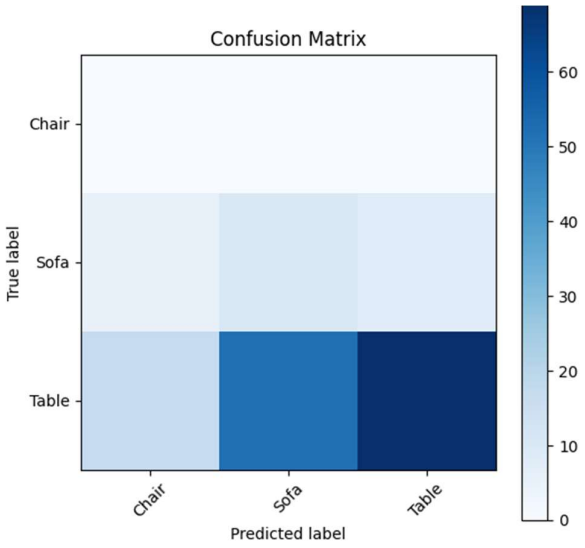


Fig 12:confusion matrix



CONCLUSION:

In conclusion, the CNN model demonstrates a strong ability to fit the training data, achieving perfect accuracy by the 16th epoch. However, the validation accuracy remains inconsistent and does not improve steadily, peaking at 68.94% and indicating overfitting. The model's performance on the validation set, where it struggles to generalize, highlights the importance of techniques like regularization, dropout, and data augmentation to improve its ability to generalize to unseen data. Further evaluation on a separate test set would offer a clearer understanding of the model's real-world performance. To improve the model's robustness, addressing overfitting should be a priority moving forward.