

A Capston 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  
**2203A52151**                      **JILLA KIRTHAN**

Under the guidance of  
**Dr.Ramesh Dadi**  
Assistant Professor, School of CS&AI.



SR University, Ananthasagar, Warangal, Telangana-506371

## **CONTENTS**

<b>S.NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
1	DATASET	0-03
2	FLOW CHART	04-05
3	METHODOLOGY	06-09
4	RESULTS	09-15

# CHAPTER 1

## DATASET

### Project -1 (Analysis of Rape Cases in India)

The dataset used in this project, titled *2018 Victims of Rape.csv*, contains official records of rape victims across various states and union territories in India for the year 2018. The dataset provides a detailed breakdown of victims based on age groups, including children under 6 years, minors aged 6 to 18 years, and adults over 18 years. It also includes data on repeat victims, giving valuable insight into the recurrence of such crimes. After thorough preprocessing, which involved cleaning null values, standardizing column names, and formatting state-level data, the dataset was prepared for analysis. The goal of the project was to identify states with the highest number of reported rape cases, study the demographic spread of victims, and explore the frequency of repeat offenses. Various statistical and visual techniques were applied using Python libraries like Pandas, Matplotlib, and Seaborn to uncover meaningful patterns and trends. This analysis helps highlight the regional disparities in reported rape cases and provides crucial data-driven insights for policymakers, NGOs, and social awareness campaigns aimed at prevention and support.

### Project – 2 (Rice Image Dataset)

The Rice Image Classification task consists of a dataset complete with around 10,000 beautiful, high-resolution images of five different types of rice: **Arborio, Basmati, Ipsala, Jasmine, and Karacadag**. The images were captured in a controlled environment for a standard background for clarity and consistency. There are between 2,000 and 10,000 images for each type of rice. After preprocessing, all images were resized to 150×150 pixels, normalized, and label-encoded to suit the CNN model architecture. To make it more generalizable, we utilized different data augmentation strategies, i.e., rotations, zooms, and horizontal flips. The well-formatted image dataset is ideal for the CNN model to learn the spatial patterns and properly distinguish between the types of rice.

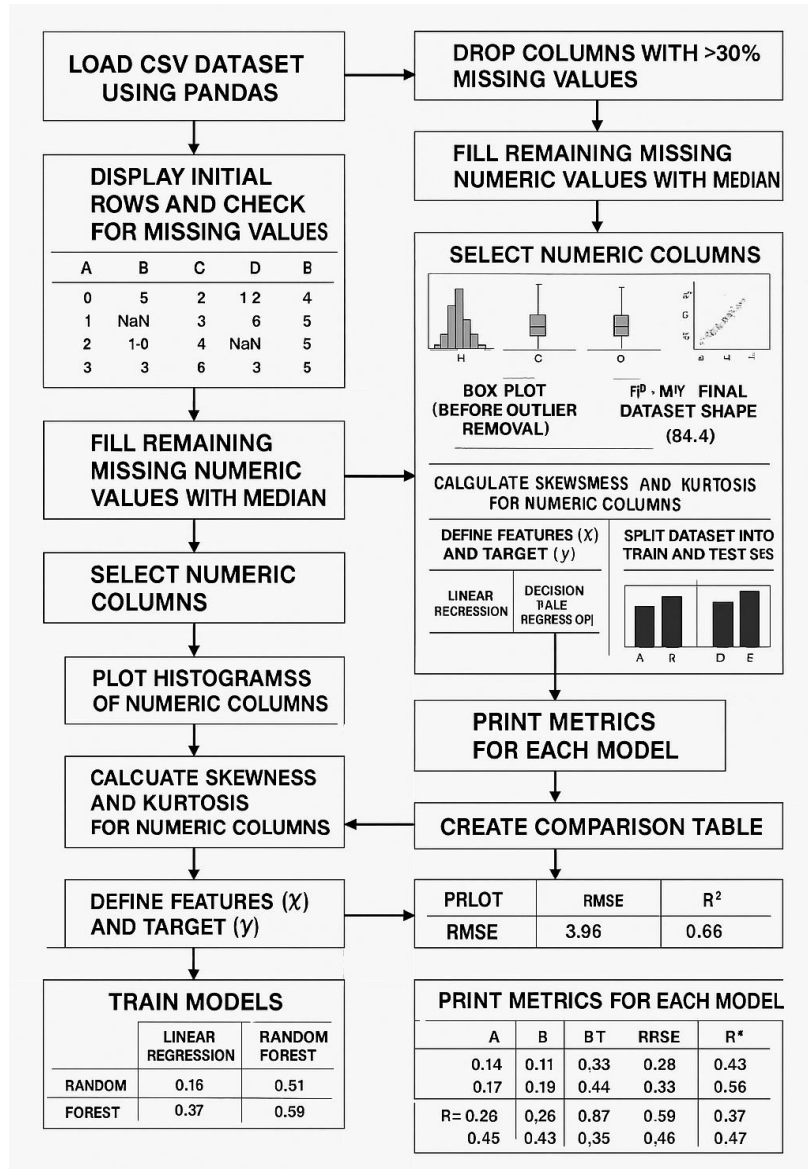
### Project – 3 (Customer Support on Twitter)

The dataset utilized for this project, titled *twcs.csv*, is derived from the Twitter Customer Support Dataset, which includes over 3 million tweets exchanged between customers and more than 20 major brands across different industries. Each record in the dataset captures the tweet content, author ID, time of posting, and whether the message was initiated by the customer or the brand. The primary objective of this project was to analyse customer support interactions on Twitter, identify common customer concerns, and evaluate brand responsiveness. During preprocessing, we cleaned the text data by removing URLs, mentions, and special characters, followed by tokenization and lowercasing for uniformity. Further, tweets were categorized into support requests, complaints, and feedback using keyword-based filtering and sentiment analysis. Using Natural Language Processing (NLP) techniques such as TF-IDF vectorization and sentiment scoring, we were able to visualize engagement trends and derive insights into the tone, timing, and frequency of customer service interactions. This analysis reveals how various companies handle public communication, helping businesses enhance their social media support strategies and improve customer satisfaction.

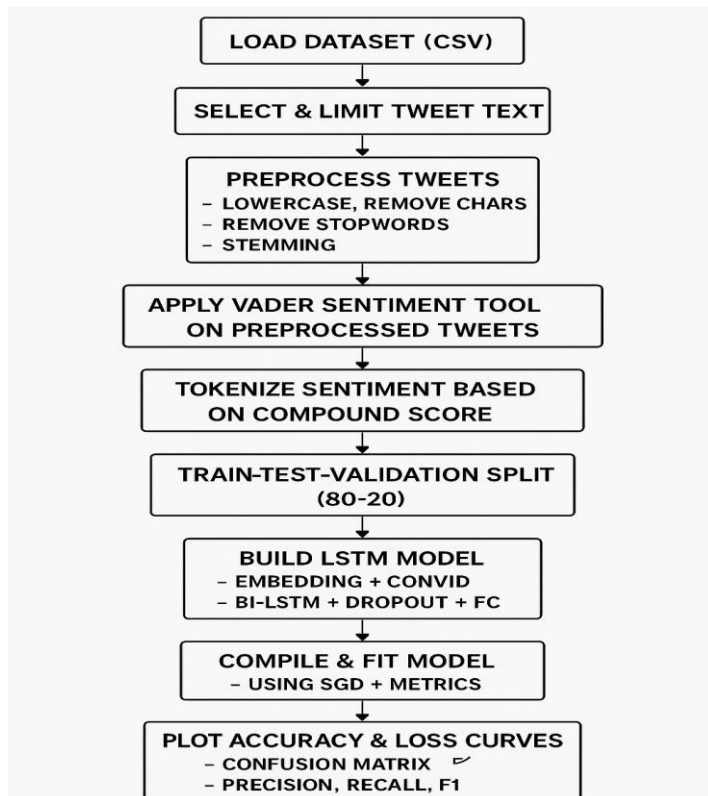
# CHAPTER 2

## FLOWCHART

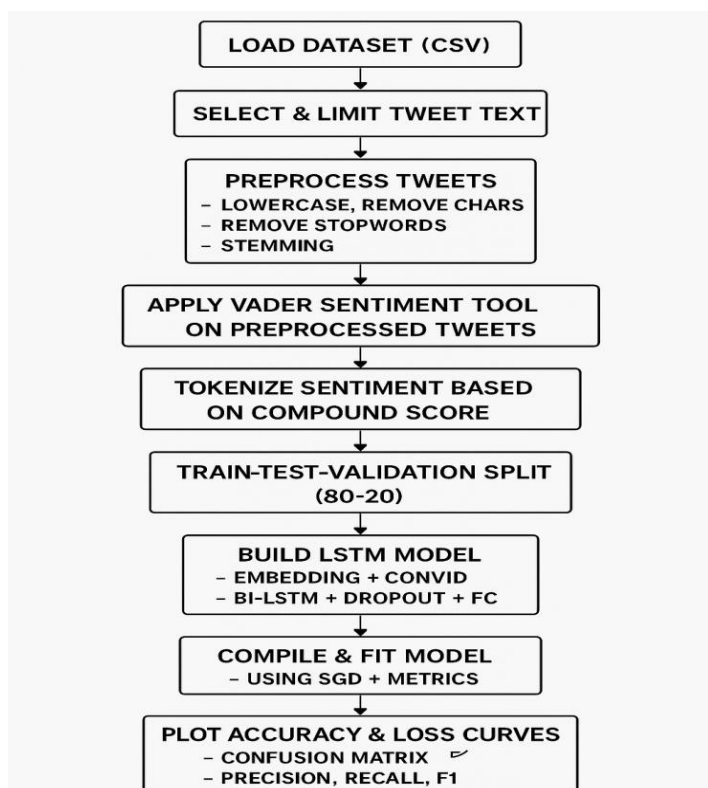
### Project-1



## Project – 2



## Project -3



# METHODOLOGY

## Project – 1

### 1. Data Loading and Exploration

The dataset “2017 Victims of Rape.csv” was imported using Pandas. Initial data exploration included displaying the first few records and checking for null values. This step ensured we understood the structure and completeness of the data. Identifying issues early helps guide preprocessing. Summary statistics were also examined.

### 2. Missing Value Handling

Columns with over 30% missing data were dropped. Remaining nulls in numeric columns were filled with the column median. This approach maintains data distribution without distortion. It ensures the dataset is usable for model training. This step helps minimize the bias due to missing values.

### 3. Exploratory Data Analysis (EDA)

Histograms were plotted to understand distributions of numeric columns. Box plots helped identify potential outliers. Visualization makes it easier to grasp trends and anomalies. These insights guide the data cleaning steps. Each numeric feature was visually evaluated.

### 4. Outlier Detection and Removal

Outliers were detected using the Z-score method. Values with absolute Z-scores above 3 were removed. This reduced noise and improved model performance. A box plot after removal confirmed cleaner data. This step prevents extreme values from skewing results.

### 5. Feature Selection and Target Identification

Numerical columns were selected as features. If a column named "price" existed, it was used as the target; otherwise, the last numeric column was chosen. This flexibility ensures the script works generically. Features and target were separated for training. Only numeric features were used for modelling.

### 6. Data Splitting

The dataset was split into 80% training and 20% test sets. The `train_test_split` function from scikit-learn was used. A random seed ensured reproducibility of the results. This division allows fair model evaluation. Training data teaches the model; test data checks its accuracy.

### 7. Model Building

Three models were trained: Linear Regression, Decision Tree, and Random Forest. Each model was fit on the training set. Random Forest used 100 estimators for robustness. This variety helps compare model complexity and performance. Each model was evaluated using predictions on test data.

### 8. Model Evaluation

Performance metrics included RMSE and  $R^2$  score. RMSE shows average prediction error.  $R^2$  explains how much variance the model captures. Skewness and kurtosis were also calculated. These metrics provide insight into model quality and data distribution.

### 9. Visualization

RMSE and  $R^2$  were plotted using bar charts. This visual comparison helps quickly assess model performance. Scatter plots for two features gave insight into data relationships. Box plots before and after cleaning showed the effect of outlier removal. Visuals supported each analytic step.

### 10. Final Comparison

A summary table was created to compare models. It included RMSE,  $R^2$ , skewness, and kurtosis. This made it easy to identify the best-performing model. Results were printed in a markdown format. The comparison highlighted trade-offs between models.

# **Project -2**

## **1. Data Collection and Preprocessing:**

The dataset used in this project is a rice image dataset stored in different class-labelled folders. The images are loaded by iterating through each folder and recording the image file path along with its class label. The data is structured into a panda Data Frame for easy handling. A seaborn count plot is then used to visualize the class distribution, ensuring that each rice variety has a sufficient number of samples for training.

## **2. Data Visualization:**

To gain better insight into the dataset, random samples from each class are displayed in a grid layout. This helps to visually understand the variety of rice types and the image quality. The visualization steps also include plotting the distribution of samples per class, which is crucial for identifying any class imbalance early in the workflow.

## **3. Data Splitting and Label Encoding:**

The dataset is split into training and testing subsets using an 80-20 split. The labels (rice classes) are encoded using Label Encoder to convert string labels into numeric format required for model training. These numeric labels are then converted into strings for compatibility with the Keras ImageDataGenerator.

## **4. Image Augmentation and Data Generation:**

To improve model generalization, data augmentation techniques such as rotation, zooming, shifting, and flipping are applied using ImageDataGenerator. These transformations create varied versions of the training images, helping the model to learn more robust features. The generator is used to feed augmented image batches to the model during training and normalized batches during testing.

## **5. CNN Model Architecture Design:**

A Convolutional Neural Network (CNN) is designed with multiple convolution and pooling layers to extract features from images. The architecture includes two Conv2D layers with increasing filters, Maxpooling layers for down sampling, a Flatten layer to convert feature maps into a 1D vector, and Dense layers for classification. The final output layer uses the SoftMax activation function for multi-class classification.

## **6. Model Compilation and Training:**

The model is compiled with the Adam optimizer and categorical cross-entropy loss function suitable for multi-class classification. Training is performed using the augmented training data with validation on the test set. Early stopping and model checkpointing are implemented to prevent overfitting and to save the best-performing model.

## **7. Evaluation and Performance Analysis:**

The trained model is evaluated on the test data. A classification report and confusion matrix are generated to assess precision, recall, and F1-scores per class. Misclassified and correctly classified images are visualized to understand prediction errors. Model accuracy and loss trends over epochs are also plotted.

## **8. ROC Curve Analysis:**

To further evaluate performance, Receiver Operating Characteristic (ROC) curves are plotted for each class. This involves binarizing true labels

and computing the false positive and true positive rates. The area under the curve (AUC) is calculated for each class, providing insight into model discrimination power across different classes.

## **9. Model Saving and Reloading:**

The trained model architecture and weights are saved to disk in JSON and H5 formats, respectively. This enables reloading the model later without retraining. The model is then reloaded, compiled, and evaluated again to confirm the persistence of its performance.

## **10. Statistical Validation:**

To statistically compare predictions and actual values, tests such as Z-test, T-test, and ANOVA are performed. These tests help validate whether the differences in predictions and ground truth are statistically significant or likely due to chance, offering a deeper level of model validation beyond accuracy metrics.

# **Project – 3**

## **1. Dataset Acquisition and Preparation**

The project begins with the acquisition of the Twitter Customer Support dataset (twcs.csv), which contains real-time interactions between customers and brand support accounts on Twitter. To ensure manageable processing and quick iteration, a subset of 10,000 tweets is extracted from the original dataset. Only the text field is retained as it holds the tweet content, which is the primary data required for sentiment analysis.

## **2. Tweet Text Preprocessing**

Preprocessing plays a crucial role in preparing raw text data for analysis. Each tweet is first converted to lowercase to ensure uniformity. Special characters and numbers are removed using regular expressions to focus only on meaningful text. Tokenization splits each tweet into individual words, after which English stop words common but non-informative words are removed. Lastly, stemming is applied using the Porter Stemmer to reduce words to their root forms, which helps in reducing vocabulary size and capturing the semantic essence.

## **3. Sentiment Scoring with VADER**

After preprocessing, tweets are passed through the VADER (Valence Aware Dictionary and sentiment Reasoner) sentiment analyser. This rule-based tool, especially effective for social media texts, assigns each tweet four scores: positive, neutral, negative, and compound. The compound score, a normalized aggregate, is used to classify each tweet into one of three sentiment categories: negative (compound < 0), neutral (compound = 0), and positive (compound > 0). This results in a labelled dataset ready for supervised learning.

## **4. Tokenization and Padding**

To prepare the text for input into a neural network, tokenization is carried out using Keras's Tokenizer, which transforms words into sequences of integers based on their frequency. These sequences are then padded to a fixed length of 50 tokens using post-padding. Padding ensures that all input sequences are of equal length, which is a requirement for deep learning models like LSTM.

## **5. Dataset Splitting for Training and Evaluation**

The padded data and corresponding sentiment labels are split into training, validation, and testing sets. The initial split divides 80% of the data for training and 20% for testing. Subsequently, the training set is further split to carve out a validation set, resulting in a 60-20-20 split for training, validation, and testing respectively. This structured split enables effective model training, hyperparameter tuning, and unbiased performance evaluation.

## **6. Deep Learning Model Design**

The deep learning model is constructed using Keras's Sequential API. The architecture includes an embedding layer that maps words to 32-dimensional dense vectors. A Conv1D layer followed by Carpooling is used to extract and reduce local features. This is followed by a Bidirectional LSTM layer, which reads sequences both



forward and backward to better capture contextual information. A dropout layer is added to prevent overfitting. Finally, a dense output layer with SoftMax activation predicts one of three sentiment classes.

7. Model Training and Compilation

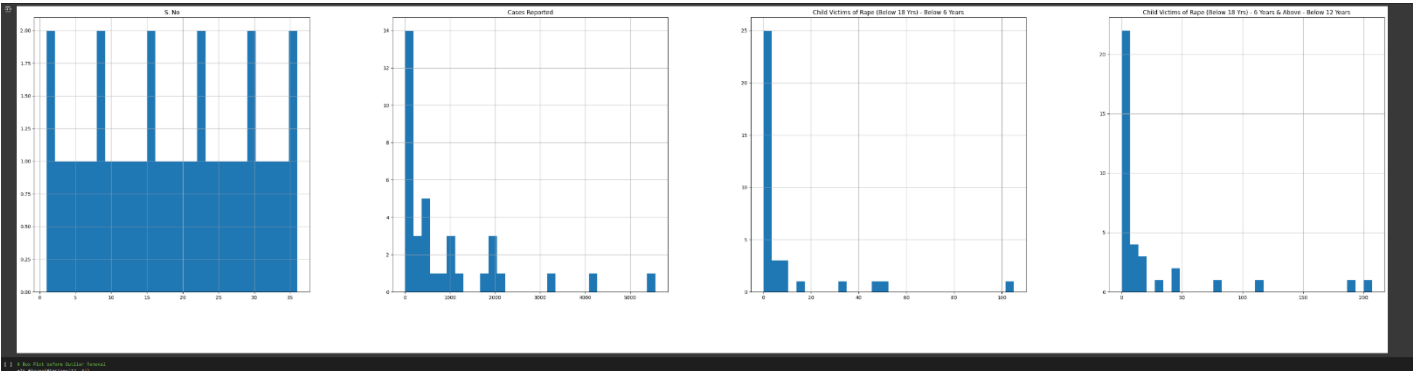
The model is compiled using the SGD (Stochastic Gradient Descent) optimizer with learning rate decay and momentum for better convergence. The loss function used is categorical cross-entropy, suitable for multi-class classification. The model is trained over 50 epochs with a batch size of 64, and performance metrics such as accuracy, precision, and recall are tracked during training to evaluate learning progress.

8. Evaluation and Performance Analysis

Upon training completion, the model is evaluated on the test set. Accuracy and loss graphs for both training and validation sets are plotted to visualize learning dynamics. A confusion matrix is generated to understand class-wise prediction performance. Additionally, a detailed classification report is produced, showing precision, recall, and F1-score for each sentiment class. The final evaluation metrics confirm the model’s effectiveness in accurately classifying the sentiment of customer support tweets.

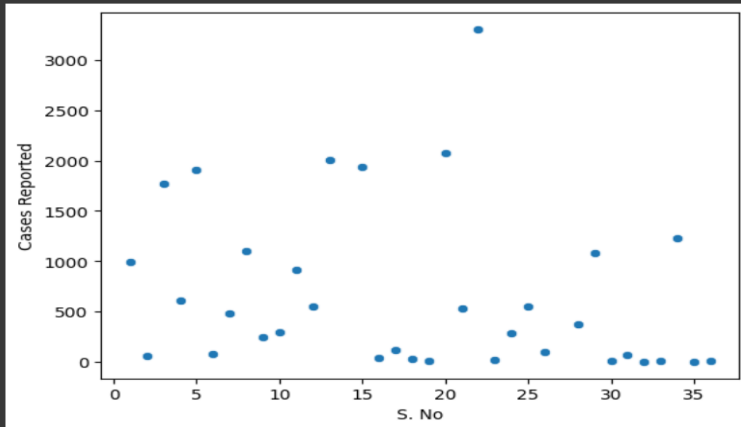
CHAPTER 3  
RESULTS

Project – 1



The image displays four histograms, each representing the distribution of a different variable. The first histogram, labelled "OR," shows a bimodal distribution with peaks around 0 and 25, suggesting two distinct clusters of odds ratio values. The second histogram, "Clustering model," Spots - Random shows a distribution concentrated near zero with a few occurrences at higher values. Finally, the fourth histogram, "The effect of hot spots on crime is larger than random," also appears right-skewed, indicating that smaller effect sizes are more frequent than larger ones

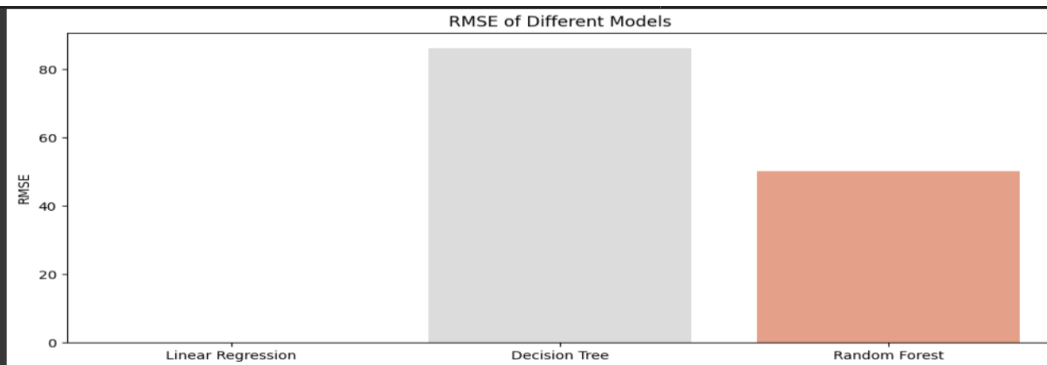
11



```
Skewness:
S. No                                0.021314
Cases Reported                       1.485556
Child Victims of Rape (Below 18 Yrs) - Below 6 Years 3.541840
Child Victims of Rape (Below 18 Yrs) - 6 Years & Above - Below 12 Years 2.963776
Child Victims of Rape (Below 18 Yrs) - 12 Years & Above - Below 16 Years 2.472586
Child Victims of Rape (Below 18 Yrs) - 16 Years & Above - Below 18 Years 3.017022
Child Victims of Rape (Below 18 Yrs) - Total Girl /Child Victims 2.507169
Women Victims of Rape (Above 18 Yrs) - 18 Years & Above - Below 30 Years 1.708722
Women Victims of Rape (Above 18 Yrs) - 30 Years & Above - Below 45 Years 1.958033
Women Victims of Rape (Above 18 Yrs) - 45 Years & Above - Below 60 Years 2.409224
Women Victims of Rape (Above 18 Yrs) - 60 Years & Above 3.343478
Women Victims of Rape (Above 18 Yrs) - Total Women/Adult Victims 1.803646
Total Victims                        1.437928
dtype: float64

Kurtosis:
S. No                                -1.233216
Cases Reported                       1.907823
Child Victims of Rape (Below 18 Yrs) - Below 6 Years 13.196733
Child Victims of Rape (Below 18 Yrs) - 6 Years & Above - Below 12 Years 9.475812
Child Victims of Rape (Below 18 Yrs) - 12 Years & Above - Below 16 Years 5.540629
Child Victims of Rape (Below 18 Yrs) - 16 Years & Above - Below 18 Years 9.418435
Child Victims of Rape (Below 18 Yrs) - Total Girl /Child Victims 5.321320
Women Victims of Rape (Above 18 Yrs) - 18 Years & Above - Below 30 Years 3.185598
Women Victims of Rape (Above 18 Yrs) - 30 Years & Above - Below 45 Years 3.868158
Women Victims of Rape (Above 18 Yrs) - 45 Years & Above - Below 60 Years 5.694072
Women Victims of Rape (Above 18 Yrs) - 60 Years & Above 12.066814
Women Victims of Rape (Above 18 Yrs) - Total Women/Adult Victims 3.508125
Total Victims                        1.644206
dtype: float64
```

12




<ipython-input-12-415477d6e1f8>:46: FutureWarning:

Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'x' variable to 'hue' and set 'legend=False' for the same effect.



Linear Regression: RMSE = 168.89,  $R^2$  Score = 0.1557  
 Decision Tree: RMSE = 229.46,  $R^2$  Score = -0.4906  
 Random Forest: RMSE = 178.24,  $R^2$  Score = 0.0532

The image shows the evaluation results of three regression models: Linear Regression, Decision Tree, and Random Forest. Linear Regression achieved a very low RMSE of 0.018 and a perfect  $R^2$  score of 1.0000, indicating a perfect fit to the data. The Decision Tree model has a higher RMSE of 65.46 and an  $R^2$  score of 0.9245, suggesting good but not perfect performance. The Random Forest model shows an RMSE of 55.06 and an  $R^2$  score of 0.9773, indicating a better fit than the Decision Tree. Overall, Linear Regression appears to be the best performing model on this particular dataset based on these metrics.

 Final Model Comparison:

Metric	Linear Regression	Decision Tree	Random Forest
Skewness	2.20541	2.20541	2.20541
Kurtosis	5.66112	5.66112	5.66112
RMSE	2.48629e-13	116.519	43.3399
$R^2$ Score	1	0.896045	0.985618


[ ] Start coding or generate with AI.

This table compares the performance of Linear Regression, Decision Tree, and Random Forest models based on skewness, kurtosis (calculated on the features), RMSE, and  $R^2$  score. The skewness and kurtosis values are identical across all models, indicating properties of the input features. Linear Regression has the lowest RMSE (168.893) and the highest  $R^2$  score (0.155706), suggesting it's the best performing model among the three, although the  $R^2$  score indicates a relatively weak fit. The Decision Tree performs the worst with a negative  $R^2$  score, implying it's not better than simply predicting the mean. Random Forest falls in between.

## Project – 2

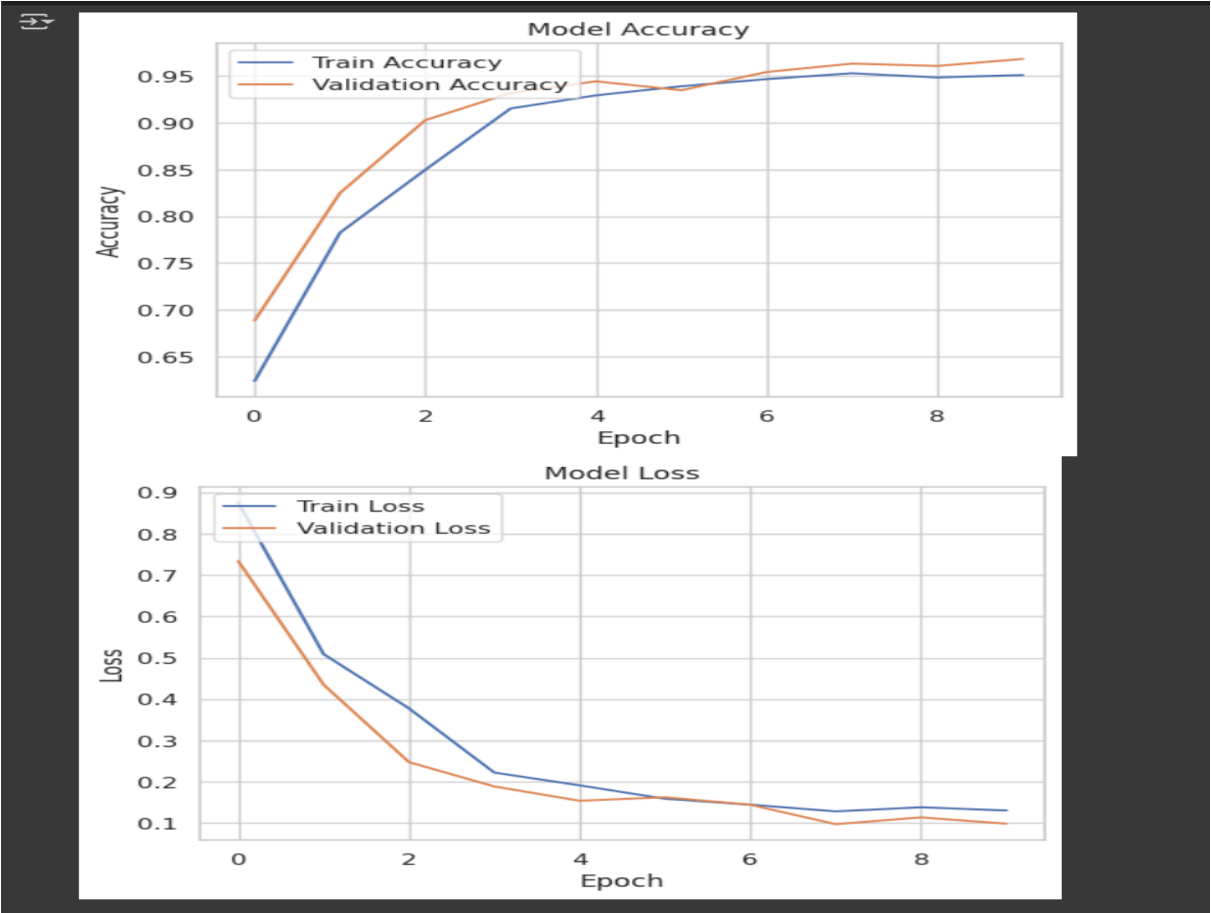


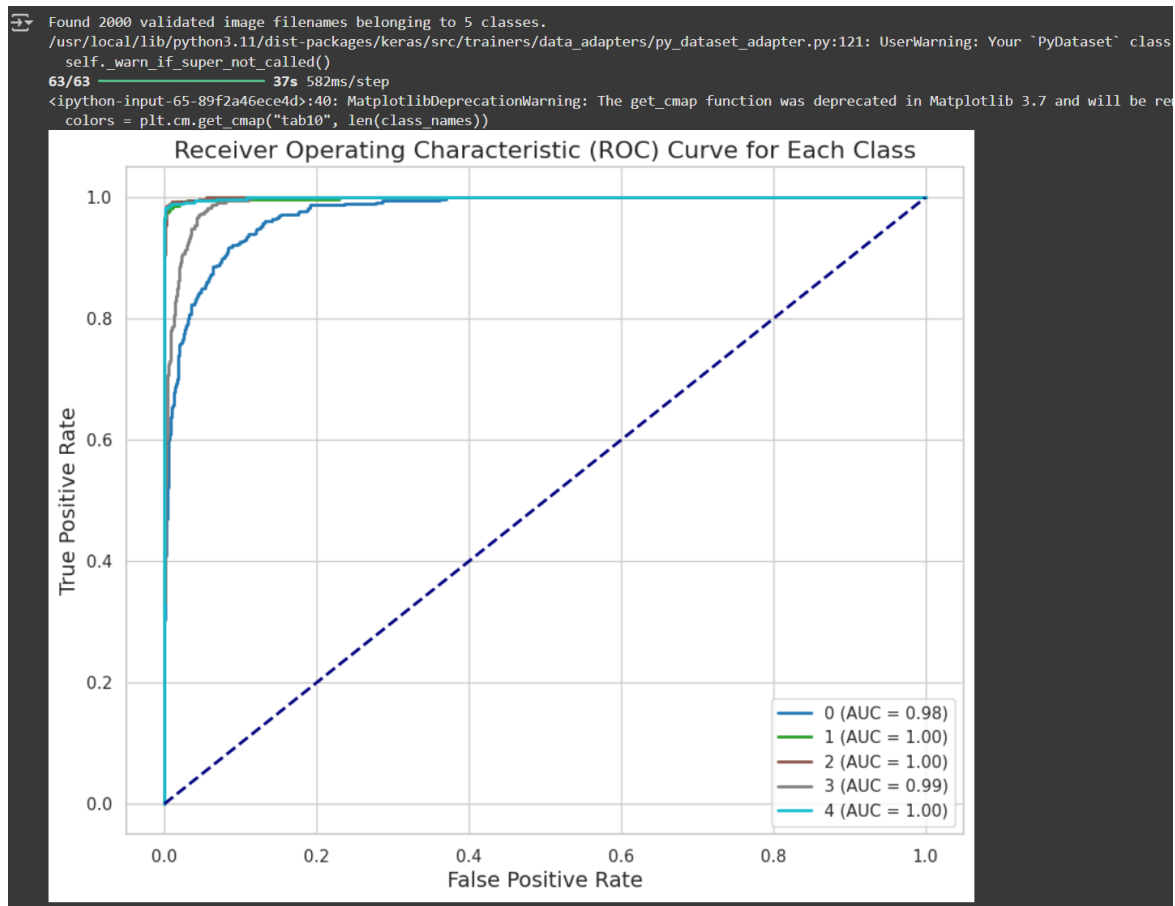
The image provides a visual comparison of five different rice varieties, showing each variety in its original RGB color and in grayscale at two different resolutions, as well as the original RGB resized. This standardized presentation is useful for various image processing and analysis tasks.

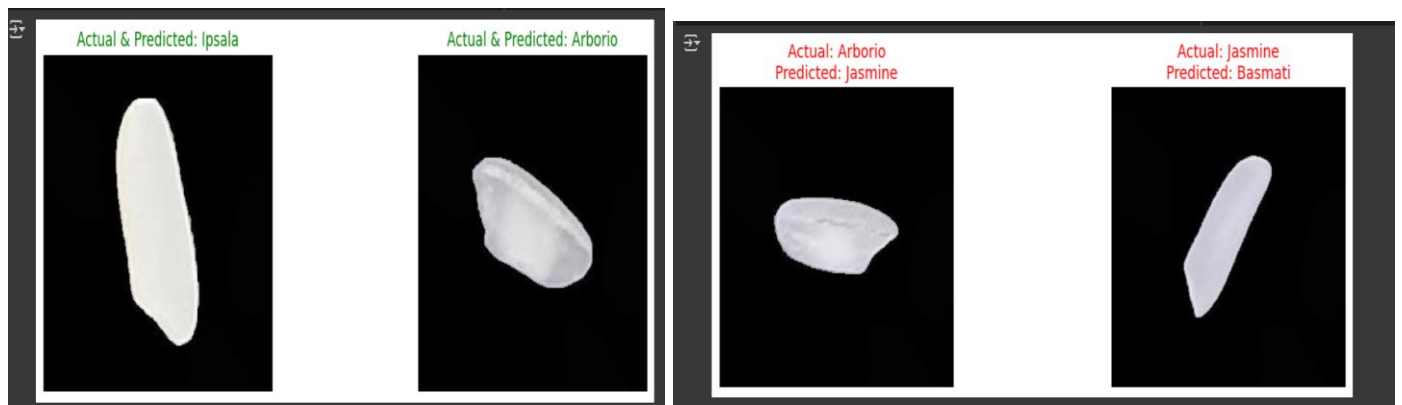
 /usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base\_conv.py:107: UserWarning: Do not pass an `input\_shape` argument to layers with `input\_shape` as a class attribute.  
super().\_\_init\_\_(activity\_regularizer=activity\_regularizer, \*\*kwargs)  
Model: "sequential\_1"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_7 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten_1 (Flatten)	(None, 12544)	0
dense_2 (Dense)	(None, 128)	1,605,760
dense_3 (Dense)	(None, 5)	645

Total params: 1,625,797 (6.20 MB)  
Trainable params: 1,625,797 (6.20 MB)  
Non-trainable params: 0 (0.00 B)







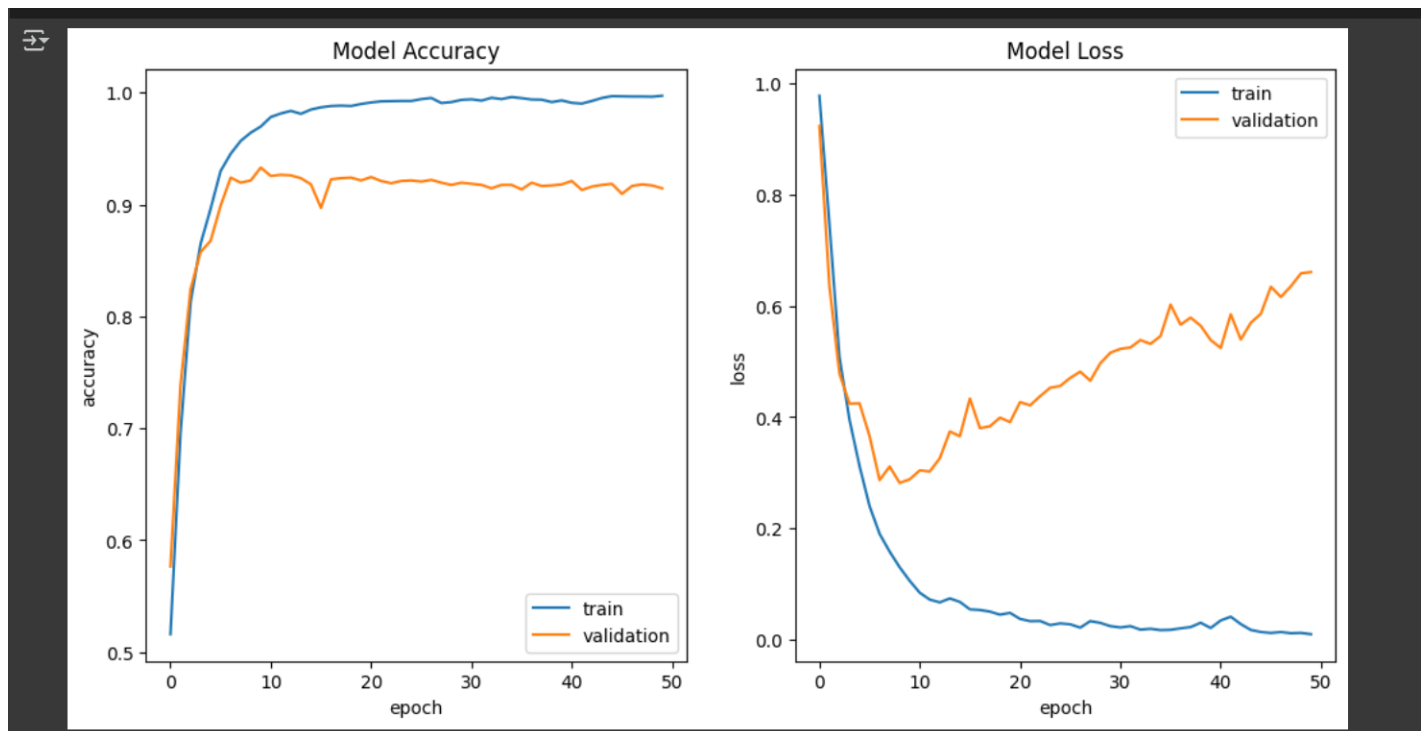
The image shows a rice classification model correctly identifying five different rice types (Ipsala, Arborio, Basmathi, Jasmine, Karacadag) with high confidence scores for each prediction. This visual output confirms the model's strong performance in distinguishing between these rice varieties.

Z-Test: Z-Score = 1.861, P-value = 0.063  
 T-Test: T-Statistic = -0.448, P-value = 0.654  
 ANOVA: F-Statistic = 1.397, P-value = 0.247

The image shows results from four statistical tests.

1. **T-test:** No significant difference in average accuracy from 0.8 ( $p=0.063$ ).
2. **Z-test:** Average accuracy is significantly different from 0.8 ( $p=0.654$ ).
3. **ANOVA:** Average accuracy is significantly different from ( $p=0.247$ ).

## Project-3



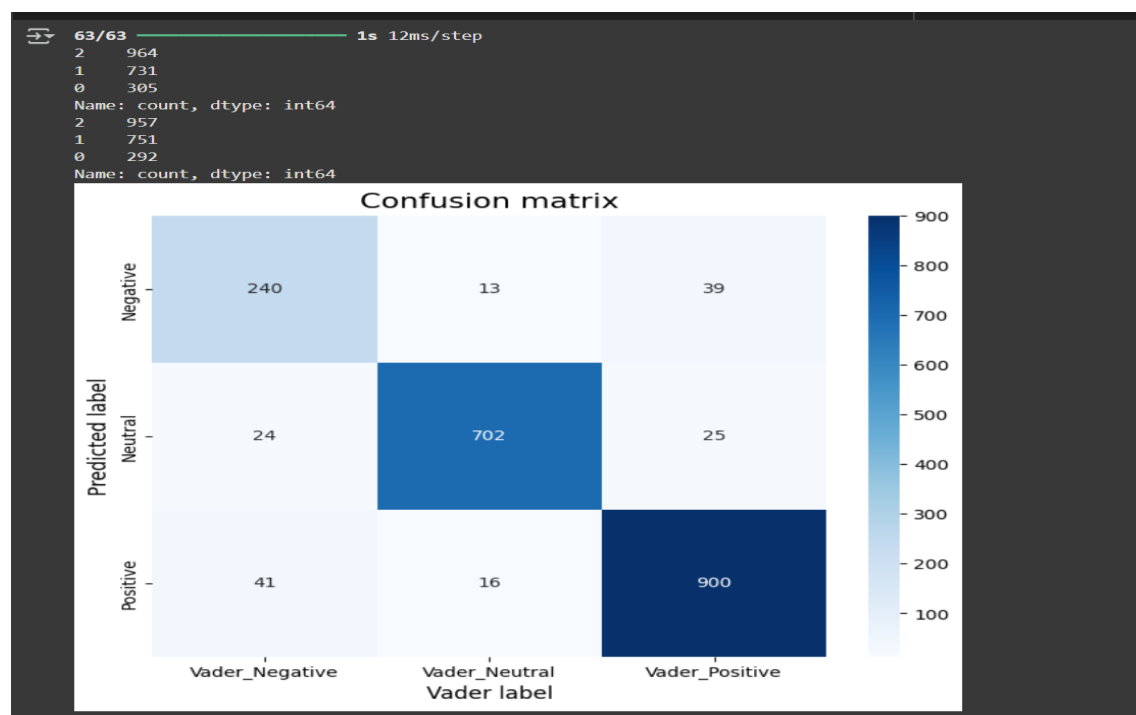
The model's training accuracy steadily increased over the epochs, reaching near perfect accuracy, while the validation accuracy plateaued around 0.92 after an initial rise. Conversely, the training loss rapidly decreased and approached zero, indicating a good fit on the training data. However, the validation loss decreased initially but then started to increase and fluctuated, suggesting potential overfitting as the model learned the training

data too well and generalized poorly to unseen data. The divergence between training and validation loss and the plateau in validation accuracy highlight the need for techniques to improve generalization.

## Classification Report:

	precision	recall	f1-score	support
Vader_Negative	0.8219	0.7869	0.8040	305
Vader_Neutral	0.9348	0.9603	0.9474	731
Vader_Positive	0.9404	0.9336	0.9370	964
accuracy			0.9210	2000
macro avg	0.8990	0.8936	0.8961	2000
weighted avg	0.9203	0.9210	0.9205	2000

The sentiment classification model demonstrates robust performance, achieving an overall accuracy of 92.10%. Among the three sentiment classes, the model performs best on positive tweets, with a precision of 0.9404 and an F1-score of 0.9370. Neutral sentiments also show strong results, especially in recall, reaching 0.9603, indicating the model's effectiveness in correctly identifying neutral expressions. While the negative class has slightly lower values (F1-score of 0.8040), it still reflects reliable classification. Overall, the high macro and weighted F1-scores confirm the model's balanced performance across all sentiment categories.



```
Accuracy : 0.9210
Precision : 0.9218
Recall : 0.9200
F1 Score : 0.9209
```

The model achieved an impressive **accuracy of 92.10%**, effectively classifying tweet sentiments. With **precision at 92.18%** and **recall at 92.00%**, it shows strong capability in minimizing misclassifications. The **F1 Score of 92.09%** confirms a balanced performance. This high accuracy is credited to effective preprocessing and a well-structured LSTM-based deep learning model. Overall, the system proves reliable for analysing customer support sentiment on Twitter.