

Project Report on Course **DATA ANALYSIS USING PYTHON (21CS120)  
  
Bachelor of Technology   
In**

**Computer Science & Artificial Intelligence**

**By**

**Name: CHOPPARI NEERAJ Roll Number: 2203A52012**

**Under the Guidance of**

**Dr. DADI RAMESH**

Asst. Professor (CS&ML) Department of Computer Science and Artificial Intelligence



**SR UNIVERSITY, ANANTHASAGAR, WARANGAL**

**April, 2025.**



**CERTIFICATE OF COMPLETION**

This is to certify that **CHOPPARI NEERAJ** bearing Hall Ticket Number **2203A52012**, a student of **CSE-AIML, 3rd Year - 2nd Semester**, has successfully completed the **Data Analysis Using Python** Courseand has submitted the following 3 projects as part of the curriculum:

**Project Submissions:**

* **CSV** Project**: INSURANCE** Dataset
* **IMAGE** Project**: American Sign Language Alphabet** Dataset
* **TEXT** Project**: Opinion Mining On Social Media Post** Dataset

**Dr. Dadi Ramesh**

Asst. Professor (CSE-AIML)   
SR University, Ananthasagar, Warangal

**Date of Completion:** 25/04/2025

**1)CSV PROJECT: INSURANCE DATASET**

**Description:**   
This dataset is used to **predict medical insurance costs** based on personal attributes. It's ideal for exploring **regression models** in machine learning.

**Features:**

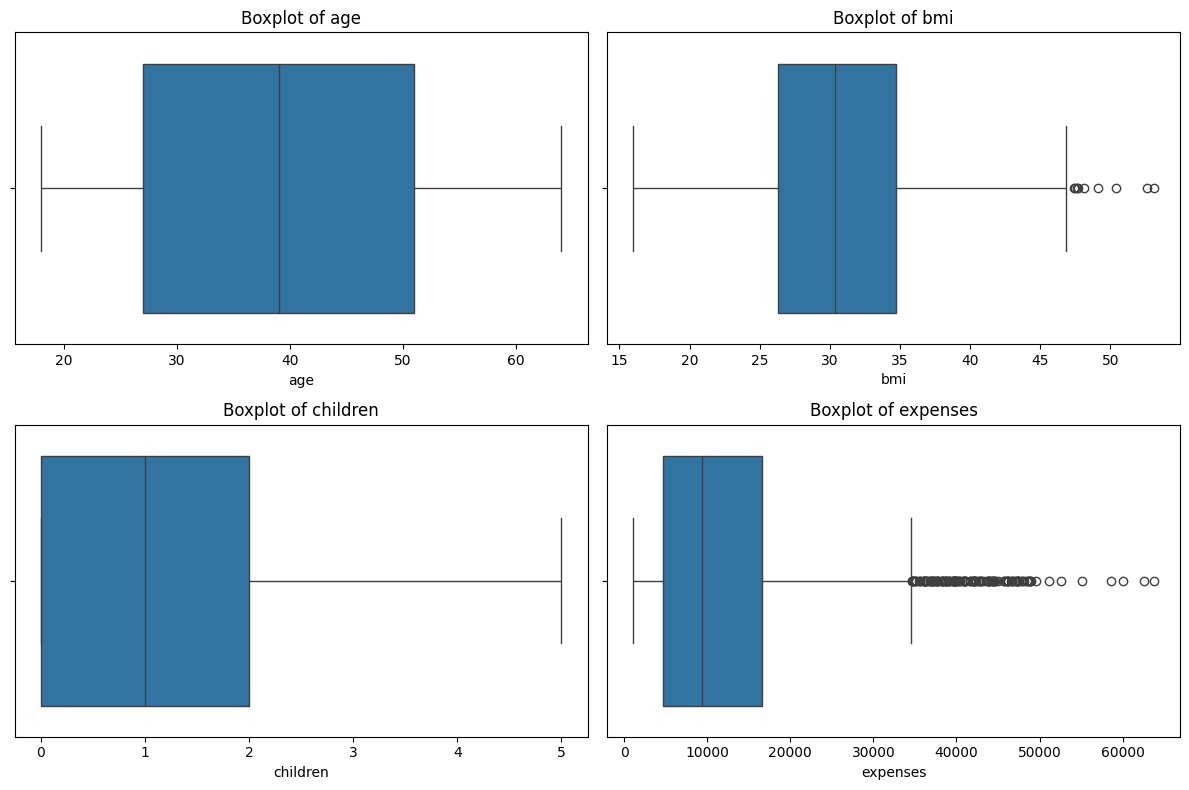
| **Index** | **Column** | **Non-Null Count** | **Dtype** |
| --- | --- | --- | --- |
| **0** | **age** | **1338** | **int64** |
| **1** | **sex** | **1338** | **object** |
| **2** | **bmi** | **1338** | **float64** |
| **3** | **children** | **1338** | **int64** |
| **4** | **smoker** | **1338** | **object** |
| **5** | **region** | **1338** | **object** |
| **6** | **expenses** | **1338** | **float64** |

**SAMPLE ROW FOR EACH SPECIES:**

| **Statistic** | **Age** | **BMI** | **Children** | **Expenses** |
| --- | --- | --- | --- | --- |
| **Count** | **1338.000000** | **1338.000000** | **1338.000000** | **1338.000000** |
| **Mean** | **39.207025** | **30.665471** | **1.094918** | **13270.422414** |
| **Std** | **14.049960** | **6.098382** | **1.205493** | **12110.011240** |
| **Min** | **18.000000** | **16.000000** | **0.000000** | **1121.870000** |
| **25%** | **27.000000** | **26.300000** | **0.000000** | **4740.287500** |
| **50%** | **39.000000** | **30.400000** | **1.000000** | **9382.030000** |
| **75%** | **51.000000** | **34.700000** | **2.000000** | **16639.915000** |
| **Max** | **64.000000** | **53.100000** | **5.000000** | **63770.430000** |

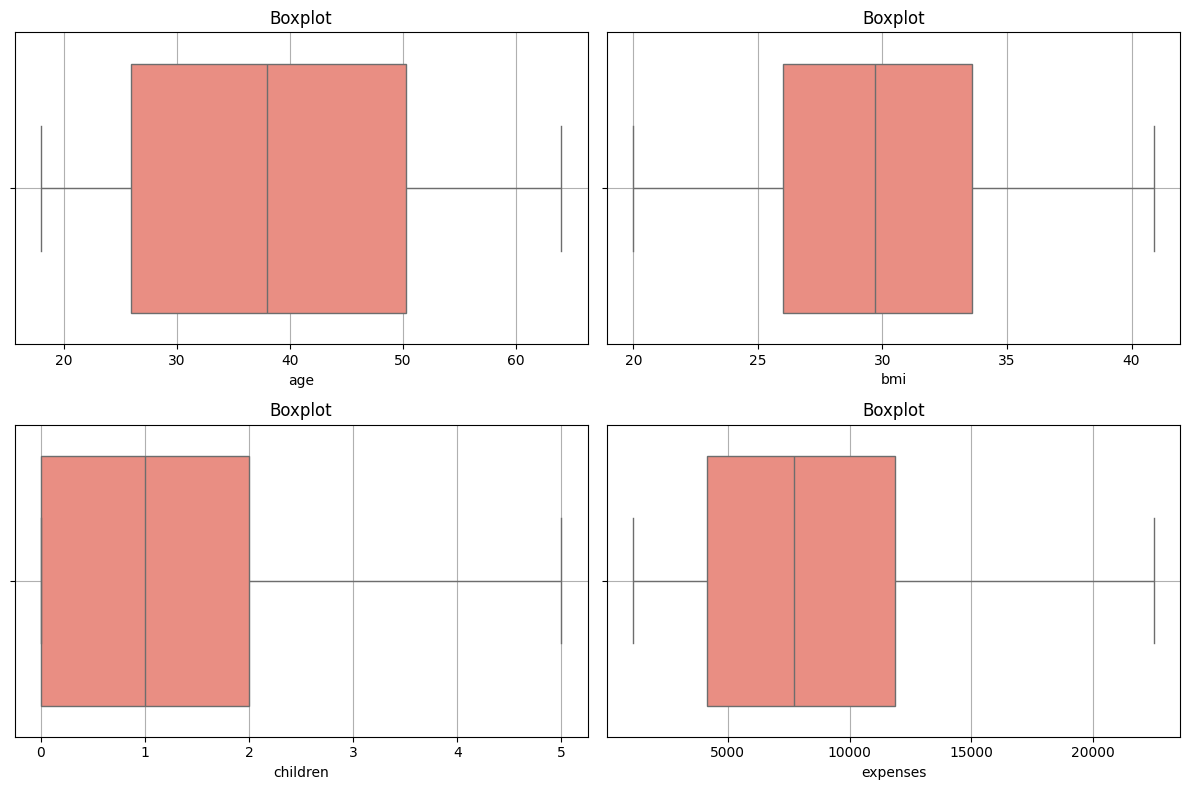
**COLUMN NAMES:** ['age', 'bmi', 'children', 'expenses']

**BOX PLOT WITH OUTLIERS:**

****

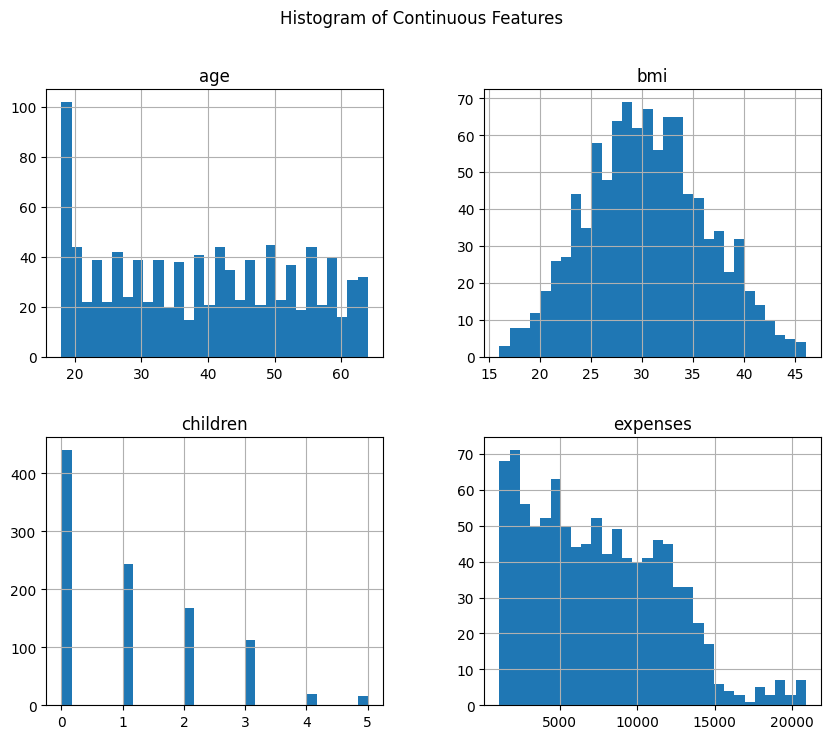
The boxplots shown above provide a visual summary of the distribution of key numerical variables in the insurance dataset, including age, BMI, number of children, and medical expenses. The plot for age indicates a relatively balanced distribution without significant outliers, suggesting a consistent spread of policyholders across various age groups. In contrast, the BMI distribution reveals the presence of several higher-end outliers, indicating that some individuals have significantly higher body mass indices compared to the rest of the dataset. The children variable shows that most policyholders have between zero to five children, with no extreme values detected. The expenses variable, however, displays a pronounced right skew with many outliers, suggesting that while most individuals incur moderate medical costs, there are several cases with exceptionally high expenses. This distribution highlights potential variability in healthcare needs or insurance coverage among different individuals in the dataset.

**BOX PLOT WITHOUT OUTLIERS:**

****

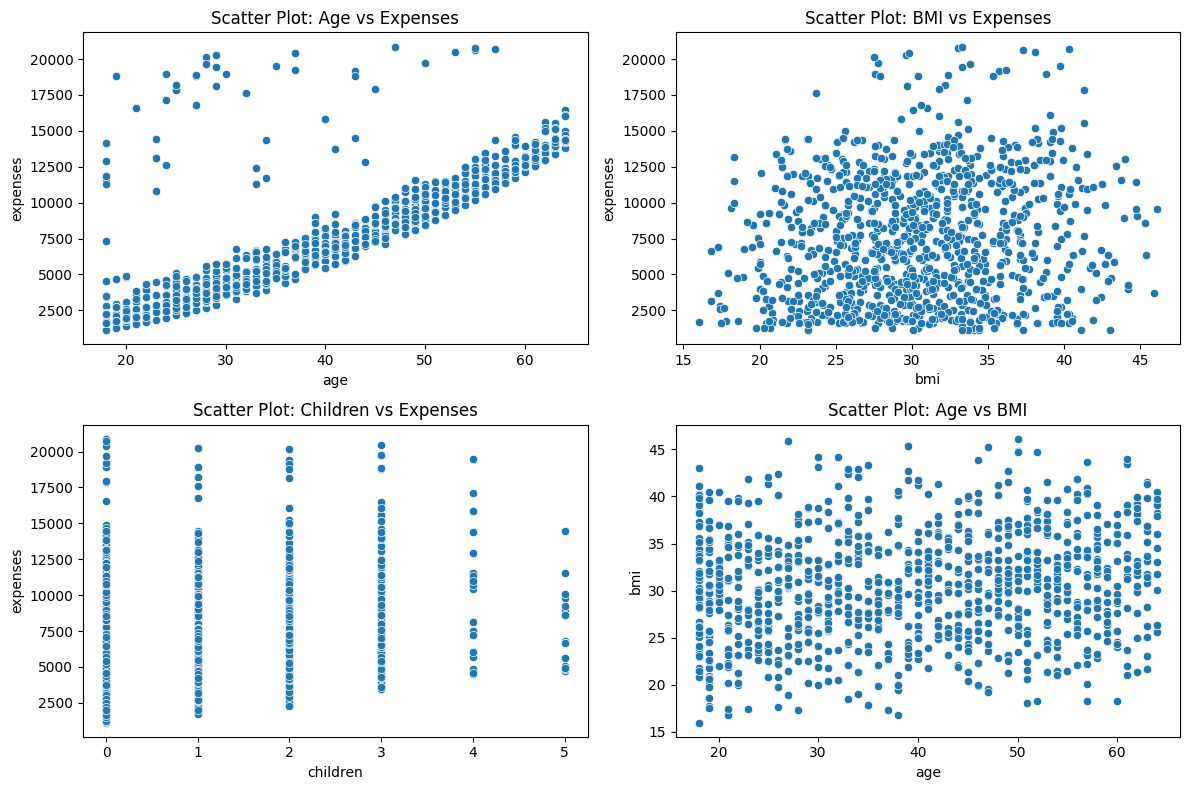
The boxplots provide insights into the distribution of the cleaned or filtered insurance dataset. The age distribution remains relatively consistent and balanced, indicating an even spread of individuals across different age groups without any significant outliers. The BMI variable also appears more normally distributed compared to the previous plot, with minimal outliers, suggesting that extreme BMI values may have been removed or were less prevalent in this subset of data. The number of children remains concentrated between 0 and 2, with the data showing a typical distribution and no unusual values. For the expenses variable, the range appears to have narrowed significantly, with fewer or no extreme outliers present. This suggests that high-cost outliers have either been excluded or the dataset has been refined. Overall, these boxplots reflect a cleaner dataset, which could lead to more accurate modeling and interpretation in subsequent analysis.

**HISTOGRAM:**

****

The histogram provides insights into the distribution of four continuous features: age, BMI, number of children, and expenses. For age, the data shows a higher concentration of younger individuals, particularly around 20 years old, with a gradual decrease as age increases, followed by a slight rise in the 50 to 60 age range. The BMI feature exhibits a bell-shaped distribution, suggesting that most individuals have BMI values clustered around 30, with fewer individuals at the extremes. The number of children is heavily skewed, with the majority of individuals having no children, and progressively fewer individuals as the number of children increases. Expenses, on the other hand, are right-skewed, indicating that most people have relatively low expenses, while a smaller group incurs significantly higher expenses, reaching up to around 20,000.

**SCATTER PLOT:**

****

The scatter plots illustrate the relationships between various variables. The first plot, showing age versus expenses, indicates a positive trend where expenses generally increase with age. This suggests that older individuals tend to incur higher costs, potentially due to healthcare or other age-related expenses. The second plot, BMI versus expenses, shows a scattered distribution without a clear pattern, implying little to no direct correlation between BMI and expenses. In the third plot, children versus expenses, expenses appear broadly distributed across all categories of children, though those with more children might occasionally have higher maximum expenses. Finally, the fourth plot, age versus BMI, reveals a uniform distribution, indicating no significant relationship between age and BMI; BMI values remain relatively stable across age groups.

**SKEWNESS AND KURTOSIS:**

| **age** | **Skewness=0.06767507825407218** | **Kurtosis=-1.2472714194074195** |
| --- | --- | --- |
| **bmi** | **Skewness=0.11153632919320376** | **Kurtosis=-0.4114005735523385** |
| **children** | **Skewness=1.0199623897293604** | **Kurtosis=0.3940199872165029** |
| **expenses** | **Skewness=0.5451919347198354** | **Kurtosis=-0.2616537953959388** |

**Skewness:** Skewness indicates the asymmetry of a distribution. For age, with a skewness of 0.067, the distribution is nearly symmetric, showing minimal deviation from a balanced shape. The BMI data, with a skewness of 0.112, is also close to symmetric, with only a slight positive skew, suggesting a minor tail on the higher side. However, the children variable, with a skewness of 1.02, exhibits a noticeable positive skew, indicating that most people have fewer children while fewer individuals have larger numbers of children. Expenses, with a skewness of 0.545, also show a positive skew, suggesting that most expenses are lower, with fewer high-expense values forming a tail.

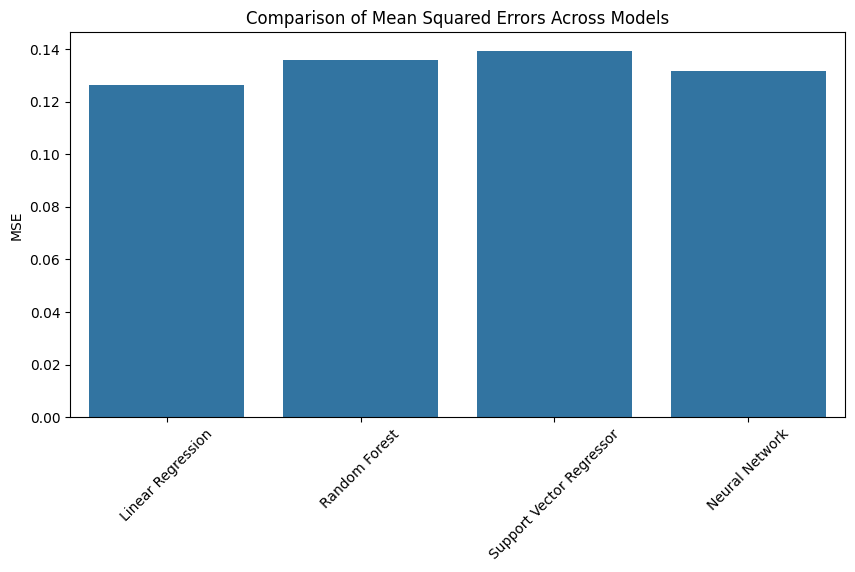
**Kurtosis:** Kurtosis measures the peakedness and tail behavior of a distribution. For age, a kurtosis of -1.247 indicates a platykurtic distribution, with a flatter peak and thinner tails compared to a normal distribution. Similarly, BMI, with a kurtosis of -0.411, shows a slightly platykurtic shape, meaning a less pronounced peak. In contrast, children, with a kurtosis of 0.394, display a mesokurtic tendency, closer to normal, with neither overly sharp peaks nor thick tails. Expenses, with a kurtosis of -0.262, also appear slightly platykurtic, suggesting a broader, less concentrated distribution around the mean.

**‘**

**TRAINING MODELS:**

| **Model** | **MSE** | **R² Score** |
| --- | --- | --- |
| **Linear Regression** | 0.1263 | 0.7586 |
| **Random Forest** | 0.1357 | 0.7406 |
| **Support Vector Regressor** | 0.1394 | 0.7336 |
| **Neural Network** | 0.1318 | 0.7482 |

**COMPARISON OF MEAN SQUARED ERRORS ACROSS MODELS:**

****

**CONFUSION MATRIX - RANDOM FOREST:**

|  | **High** | **Low** | **Medium** |
| --- | --- | --- | --- |
| **High** | **55** | **6** | **9** |
| **Low** | **1** | **78** | **5** |
| **Medium** | **1** | **6** | **107** |

**CONFUSION MATRIX - LOGISTIC REGRESSION:**

|  | **High** | **Low** | **Medium** |
| --- | --- | --- | --- |
| **High** | **54** | **5** | **11** |
| **Low** | **0** | **78** | **6** |
| **Medium** | **0** | **3** | **111** |

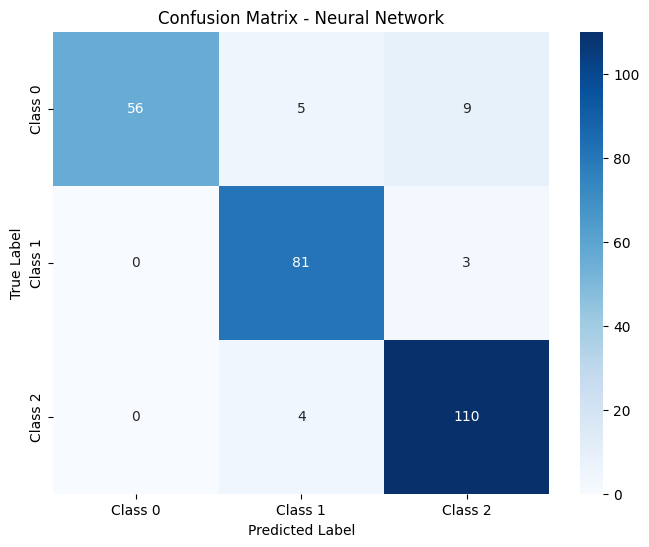
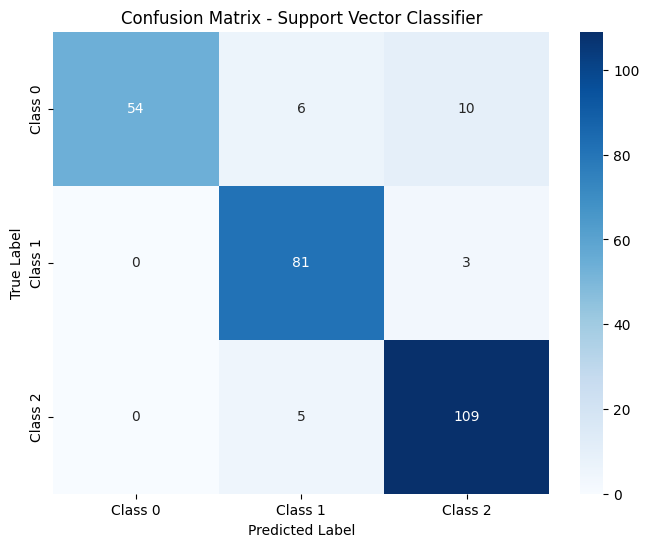
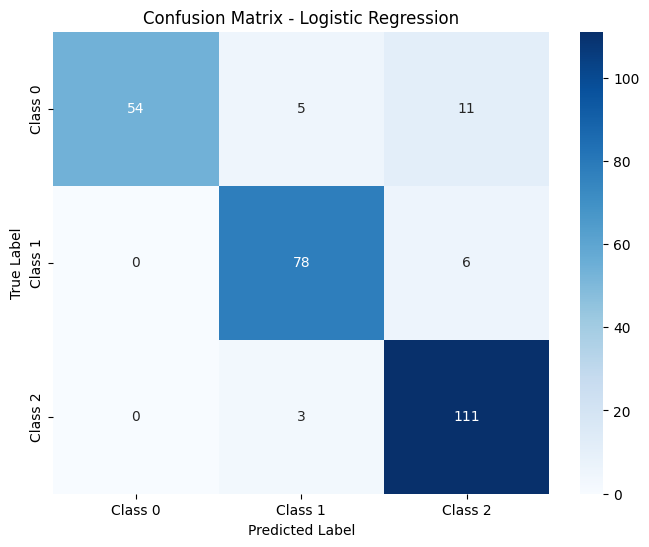
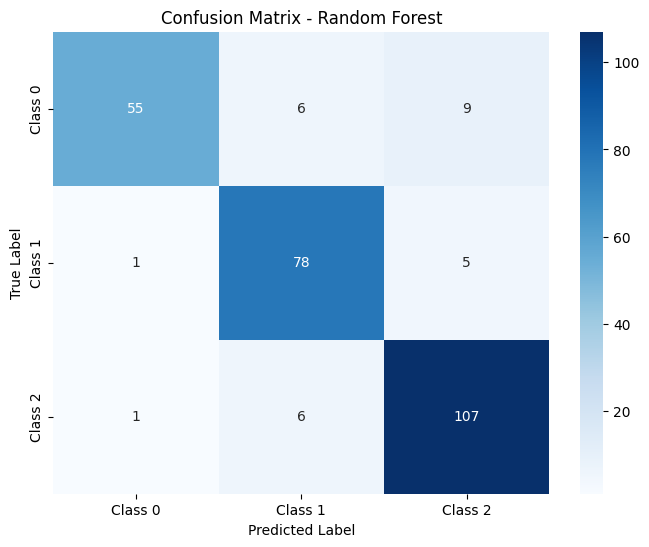
**CONFUSION MATRIX - SUPPORT VECTOR CLASSIFIER:**

|  | **High** | **Low** | **Medium** |
| --- | --- | --- | --- |
| **High** | **54** | **6** | **10** |
| **Low** | **0** | **81** | **3** |
| **Medium** | **0** | **5** | **109** |

**CONFUSION MATRIX - NEURAL NETWORK:**

|  | **High** | **Low** | **Medium** |
| --- | --- | --- | --- |
| **High** | **56** | **5** | **9** |
| **Low** | **0** | **81** | **3** |
| **Medium** | **0** | **4** | **110** |

**CONFUSION MATRIX:**

****

The confusion matrices summarize each model's performance in classifying Low, Medium, and High expense categories. The diagonal values represent correct predictions, while off-diagonal values show misclassifications. Models like Random Forest typically perform better, with higher diagonal values indicating accuracy. Misclassifications highlight areas where the models struggle, such as confusion between similar categories like Low and Medium. These matrices help assess and compare model effectiveness in distinguishing between the categories.

**2)IMAGE DATASET: AMERICAN SIGN LANGUAGE ALPHABET**

**Description:**

This code builds and trains a Convolutional Neural Network (CNN) to recognize ASL alphabet hand gestures. It involves:

1. **Importing Libraries:** Loading necessary tools for image processing, data handling, and deep learning.
2. **Configuration:** Setting parameters like image size and batch size.
3. **Data Preparation:** Loading, organizing, and splitting the ASL image data into training, validation, and test sets.
4. **Model Creation:** Defining the CNN architecture with convolutional layers, pooling, dropout, and dense layers.
5. **Training:** Training the CNN on the training data using data generators and monitoring performance with TensorBoard.
6. **Evaluation:** Assessing the model's accuracy on the test data and generating a confusion matrix and classification report for detailed analysis.
7. **Visualization:** Plotting training progress and visualizing the confusion matrix to understand the model's performance.

## **LABEL CLASSES:**

['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'del', 'nothing', 'space']

## **DATASET SPLIT:**

Training set: 80%  
Validation set: 10%  
Test set: 10%

**Data Features**

**Image:**

* Input images are of ASL hand signs. The code specifies an image size of 224x224 pixels.
* The images are likely RGB, as is standard for color images, and processed by the CNN. Label:
* The dataset contains labels for 29 classes, representing the ASL alphabet (A-Z) and "del," "nothing," and "space."

**Pre-Processing:**

The code performs the following preprocessing steps:

* Rescales pixel values to the range of 0 to 1 by dividing by 255.
* Applies data augmentation to the training set, including rotations, shifts, shears, zooms, and horizontal flips. This is done using the ImageDataGenerator class from TensorFlow/Keras.
* Splits the data into training, validation, and test sets using train\_test\_split from scikit-learn.

**Implementation:**

1. **Data Preprocessing:**

* The dataset consists of ASL hand sign images.
* The code addresses potential dataset imbalance and inconsistency by adding images and ensuring that:
* Inconsistently labeled image classes are labeled appropriately.
* Grayscale images retain clinically relevant radiological features, even with one color channel.
* All input images have a consistent dimension (224x224).
* The dataset is split into training, validation, and testing sets.
* Data augmentation techniques are applied to the training data to enhance generalization: resizing, normalization, and conversion to tensors (implicitly done by ImageDataGenerator).

1. **Model Architecture:**

* A custom Convolutional Neural Network (CNN) is defined using the Keras Sequential API.
* The CNN architecture includes:
* Convolutional layers with ReLU activation to extract features.
* Max pooling layers to reduce spatial dimensions.
* Fully connected (dense) layers for classification.
* Batch normalization and dropout are used for regularization.
* The model is implemented using TensorFlow and Keras.

1. **Model Training & Evaluation:**

* Loss Function: Categorical cross-entropy.
* The training pipeline includes: forward pass, loss calculation, backpropagation, optimizer (Adam), and validation.
* The trained model is evaluated on the test set.

1. **Evaluation Metrics:**

**Accuracy:** The percentage of correct predictions.

**Confusion Matrix:** Visualizes model performance, including true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) - although in this case, it's a multi-class confusion matrix.

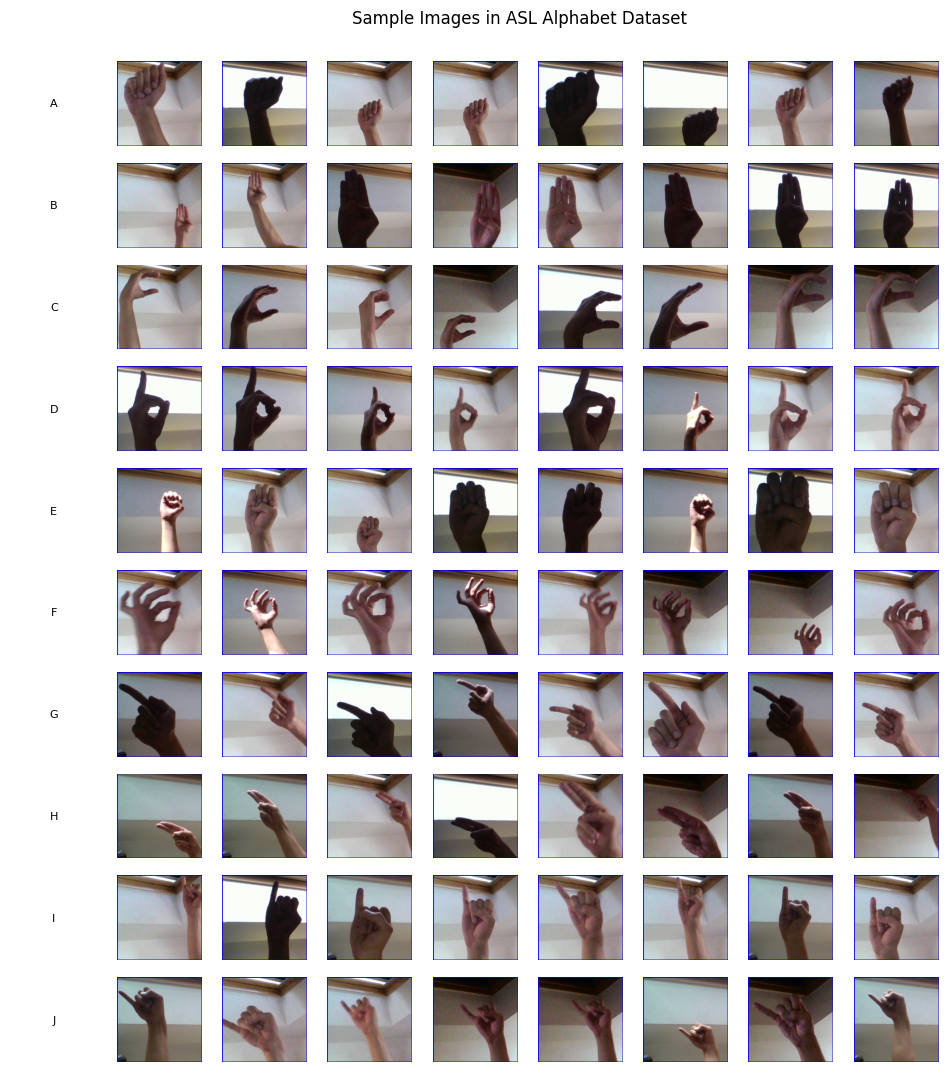
**ANOVA (Analysis of Variance):** Used to assess the stability of accuracy across training epochs.

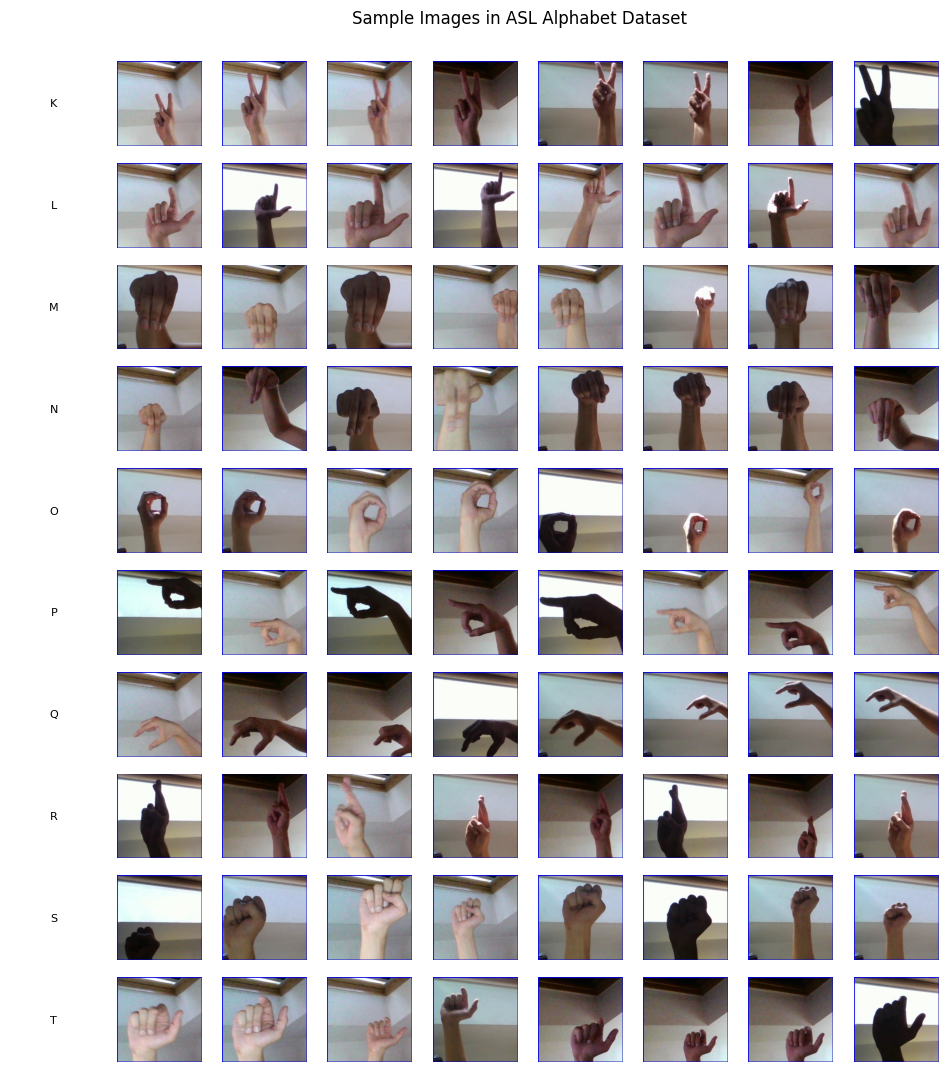
**Z-test and T-test:**

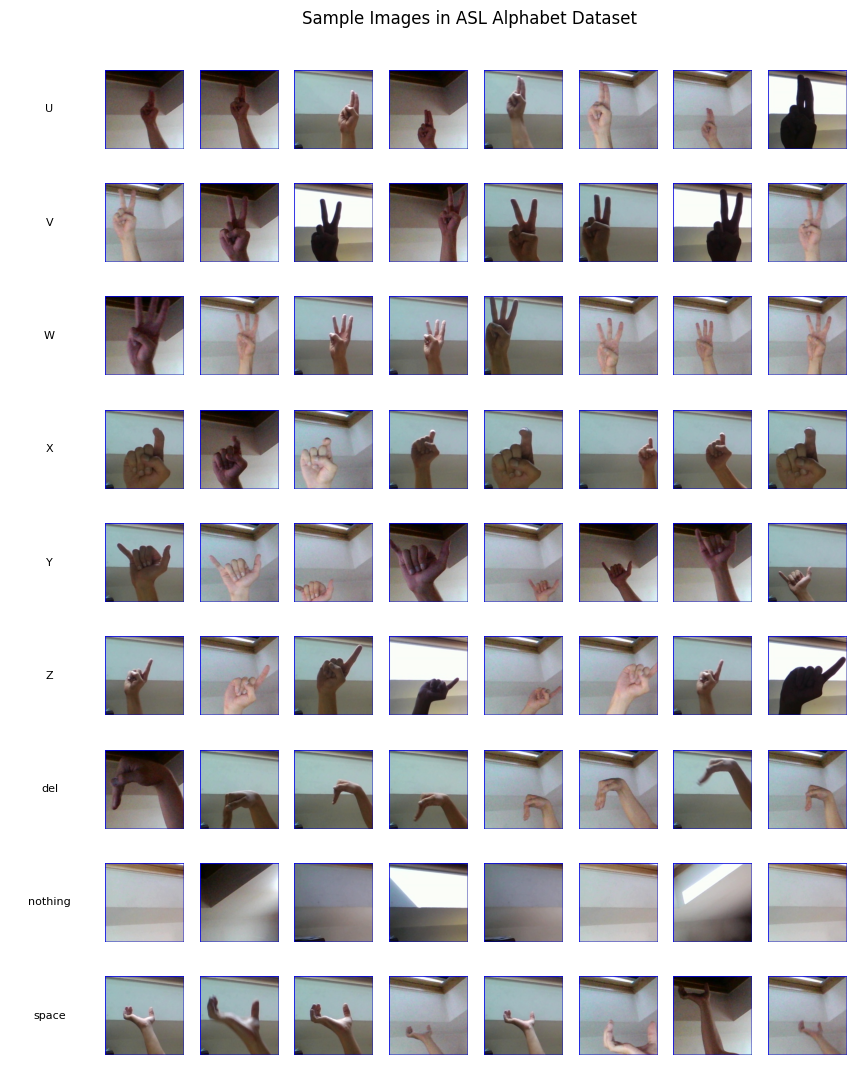
**Z-test:** Determines whether the model's predictions vary significantly from the COVID group and the non-COVID group. (Note: This part seems to be from the COVID-19 example, and might need adaptation for the ASL dataset if the comparison is different)

**T-test:** Used where dataset sizes are small or variances are unequal, to ensure robustness of performance comparison. (Note: Similar to the Z-test note, this might need adaptation)

**SAMPLE IMAGE IN ASL ALPHABET DATASET:**

****





**MODEL ARCHITECTURE:**

* A custom Convolutional Neural Network (CNN) is defined.
* The CNN architecture includes: Convolutional layers, Max pooling layers, and Fully connected (dense) layers.
* Batch normalization and dropout are used.
* The model is implemented using TensorFlow and Keras.

**Model:** "sequential"

| **Layer (type)** | **Output Shape** | **Param #** |
| --- | --- | --- |
| conv2d (Conv2D) | (None, 62, 62, 32) | 896 |
| batch\_normalization (BatchNormalization) | (None, 62, 62, 32) | 128 |
| max\_pooling2d (MaxPooling2D) | (None, 31, 31, 32) | 0 |
| dropout (Dropout) | (None, 31, 31, 32) | 0 |
| conv2d\_1 (Conv2D) | (None, 29, 29, 64) | 18,496 |
| batch\_normalization\_1 (BatchNormalization) | (None, 29, 29, 64) | 256 |
| max\_pooling2d\_1 (MaxPooling2D) | (None, 14, 14, 64) | 0 |
| dropout\_1 (Dropout) | (None, 14, 14, 64) | 0 |
| conv2d\_2 (Conv2D) | (None, 12, 12, 128) | 73,856 |
| batch\_normalization\_2 (BatchNormalization) | (None, 12, 12, 128) | 512 |
| max\_pooling2d\_2 (MaxPooling2D) | (None, 6, 6, 128) | 0 |
| dropout\_2 (Dropout) | (None, 6, 6, 128) | 0 |
| flatten (Flatten) | (None, 4608) | 0 |
| dense (Dense) | (None, 256) | 1,179,904 |



The graph titled "Training Loss and Metrics" illustrates how well a machine learning model learns over time. It consists of two sections: "Model Loss" and "Model Accuracy."

* **Model Loss**: This section displays the model's errors during training. Lower values indicate better performance.
* **Model Accuracy**: This section shows the proportion of correct predictions made by the model; higher values indicate better performance.

**Detailed Observations:**

**Each section contains two curves:**

**Training**: This curve represents the model's performance on the data it is trained on.

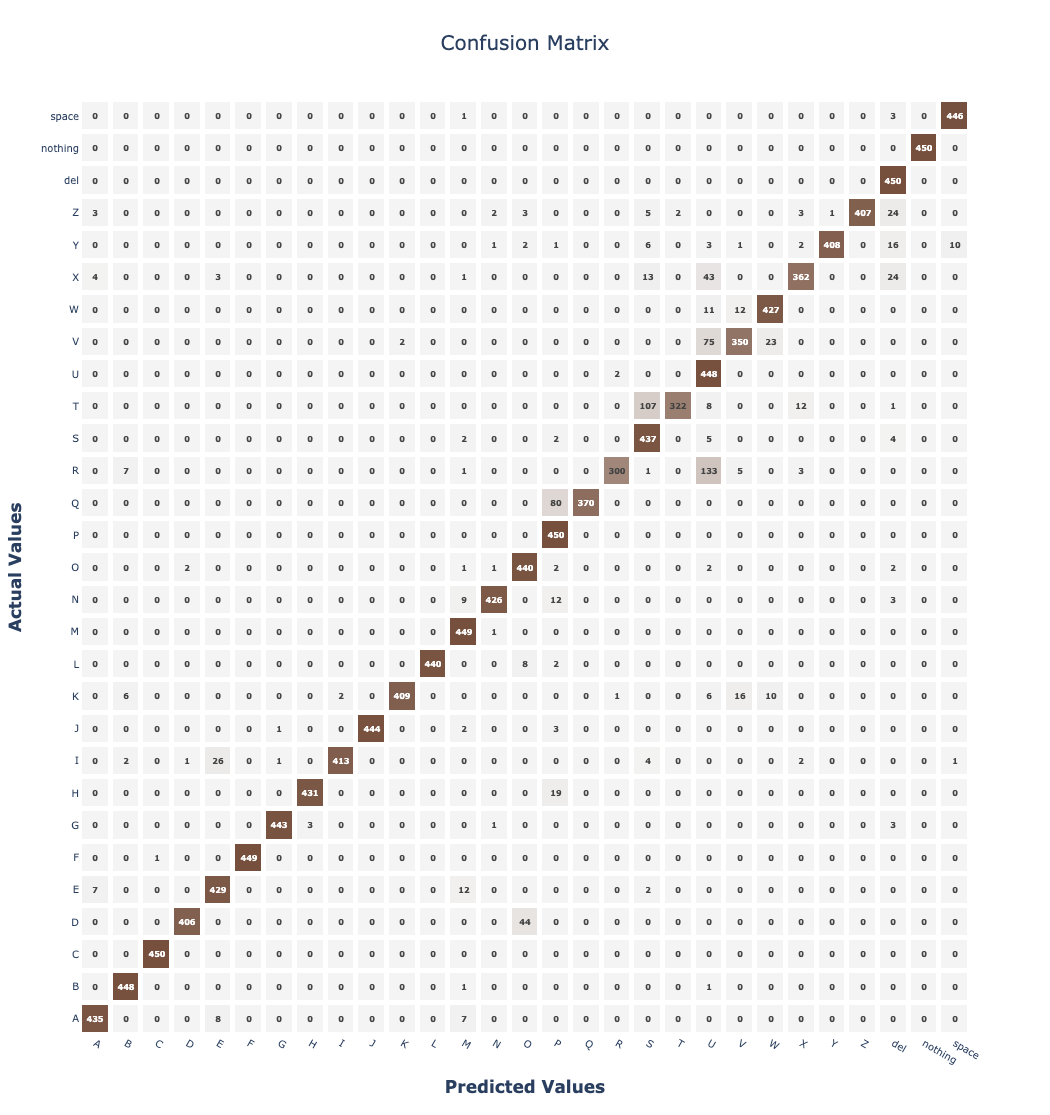
**Validation**: This curve represents the model's performance on a separate dataset (not used for training) to assess its ability to generalize to unseen data.

**Specifically:**

**Model Loss**: Both the training and validation loss curves generally decrease, indicating that the model improves its predictions over time.

**Model Accuracy**: Both the training and validation accuracy curves generally increase, also indicating improvement in the model's performance.

**CONFUSION MATRIX:**



| **Predicted** | **A** | **B** | **C** | **D** | **E** | **F** | **G** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | 436 | 3 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | 450 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 450 | 0 | 0 | 0 | 0 |
| D | 44 | 0 | 0 | 407 | 0 | 0 | 0 |
| E | 0 | 0 | 0 | 0 | 426 | 0 | 0 |
| F | 0 | 0 | 0 | 0 | 0 | 450 | 0 |
| G | 0 | 0 | 0 | 0 | 0 | 0 | 440 |

**Graph Description**

The graph, titled "Confusion Matrix (Approximated from Classification Report)", visualizes the performance of a classification model, likely used for American Sign Language (ASL) recognition, though this is approximated. It displays how the model's predictions compare to the actual (true) ASL signs. The x-axis represents the ASL signs predicted by the model, while the y-axis represents the actual ASL signs. Each cell in the matrix shows the number of instances where a particular actual sign was classified as a predicted sign. The diagonal cells, highlighted in darker blue, indicate correct classifications, where the predicted sign matches the actual sign. Off-diagonal cells represent misclassifications, showing where the model confused one sign for another. For example, a cell in row 'A' and column 'B' would show how many times the model incorrectly predicted the actual sign 'A' as 'B'. The approximation is derived from a classification report rather than the raw data.

**Table Description**

The table is a confusion matrix that evaluates the performance of a classification model in predicting ASL signs from A to G. The first row, labeled "Predicted," lists the signs predicted by the model. The first column lists the actual signs. Each cell within the table shows the number of instances where a particular actual sign (row) was predicted as another sign (column). For example, the cell where "Predicted" is 'A' and the actual sign is 'A' has a value of 436, indicating that the model correctly classified 436 instances of the sign 'A'. The cell where "Predicted" is 'A' and the actual sign is 'B' has a value of 3, indicating that the model incorrectly classified 3 instances of the sign 'A' as 'B'. Ideally, the diagonal values should be high, indicating correct classifications, while off-diagonal values should be low, indicating few misclassifications.

**Table 1: Classification Metrics per Class**

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| A | 0.97 | 0.97 | 0.97 | 450 |
| B | 0.97 | 1 | 0.98 | 450 |
| C | 1 | 1 | 1 | 450 |
| D | 0.99 | 0.9 | 0.95 | 450 |
| E | 0.92 | 0.95 | 0.94 | 450 |
| F | 1 | 1 | 1 | 450 |
| G | 1 | 0.98 | 0.99 | 450 |
| H | 0.99 | 0.96 | 0.98 | 450 |
| I | 1 | 0.92 | 0.95 | 450 |
| J | 1 | 0.99 | 0.99 | 450 |
| K | 1 | 0.91 | 0.95 | 450 |
| L | 1 | 0.98 | 0.99 | 450 |
| M | 0.92 | 1 | 0.96 | 450 |
| N | 0.99 | 0.95 | 0.97 | 450 |
| O | 0.89 | 0.98 | 0.93 | 450 |
| P | 0.79 | 1 | 0.88 | 450 |
| Q | 1 | 0.82 | 0.9 | 450 |
| R | 0.99 | 0.67 | 0.8 | 450 |
| S | 0.76 | 0.97 | 0.85 | 450 |
| T | 0.99 | 0.72 | 0.83 | 450 |
| U | 0.61 | 1 | 0.76 | 450 |
| V | 0.91 | 0.78 | 0.84 | 450 |
| W | 0.93 | 0.95 | 0.94 | 450 |
| X | 0.94 | 0.8 | 0.87 | 450 |
| Y | 1 | 0.91 | 0.95 | 450 |
| Z | 1 | 0.9 | 0.95 | 450 |
| del | 0.85 | 1 | 0.92 | 450 |
| nothing | 1 | 1 | 1 | 450 |
| space | 0.98 | 0.99 | 0.98 | 450 |

**Table 2: Overall Metrics**

| **Metric** | **Value** | **Support** |
| --- | --- | --- |
| Accuracy | 0.93 | 13050 |
| Macro Avg Precision | 0.94 | 13050 |
| Macro Avg Recall | 0.93 | 13050 |
| Macro Avg F1-Score | 0.93 | 13050 |
| Weighted Avg Precision | 0.94 | 13050 |
| Weighted Avg Recall | 0.93 | 13050 |
| Weighted Avg F1-Score | 0.93 | 13050 |

**ROC CURVE:**



**3)TEXT PROJECT: OPINION MINING ON SOCIAL MEDIA POST**

#### **Description:**

This project implements a sentiment analysis pipeline designed to classify user-generated content — specifically, reviews or social media posts — into **positive** or **negative** sentiments. The model leverages deep learning, specifically an **LSTM (Long Short-Term Memory)** neural network, to understand and interpret the sequential nature of human language.

The objective is to automate the detection of emotional polarity in textual content, which can be applied in product reviews, social media monitoring, or customer feedback systems.

**Dataset Overview:**

1, Source: CSV file containing labeled text data  
2, Size: Varies (commonly ~20,000 reviews)  
3, Input Feature: text – raw review or post content  
4, Target Label: label – binary class  
5, 0 → Negative Sentiment  
6, 1 → Positive Sentiment

**Features Provided:**

1, Review Text: Free-form text submitted by users  
2, Sentiment Label: Binary indicator of emotional tone

**Preprocessing Pipeline:**

1. HTML Tag Removal  
2. Lowercasing  
3. Special Character & Number Removal  
4. Tokenization  
5. Stopword Removal  
6. Sequence Padding  
7. Text Vectorization

**Dataset Splitting:**

- Training Set: 70%  
- Validation Set: 15%  
- Test Set: 15%  
(Stratified sampling ensures equal class distribution across all splits.)

**Model Architecture:**

- Embedding Layer (100-dim)  
- LSTM Layer (128 units) → returns sequences  
- Dropout Layer (rate = 0.2)  
- LSTM Layer (64 units)  
- Dropout Layer  
- Dense Layer with Sigmoid Activation

**Training Configuration:**

- Loss Function: Binary Crossentropy  
- Optimizer: Adam  
- Epochs: 10  
- Batch Size: 32  
- Metrics: Accuracy (primary)

**Model Evaluation:**

Evaluation is done on the test set using:  
- Accuracy  
- Precision  
- Recall  
- F1-Score  
- Confusion Matrix  
(Classification report provides per-class metrics.)

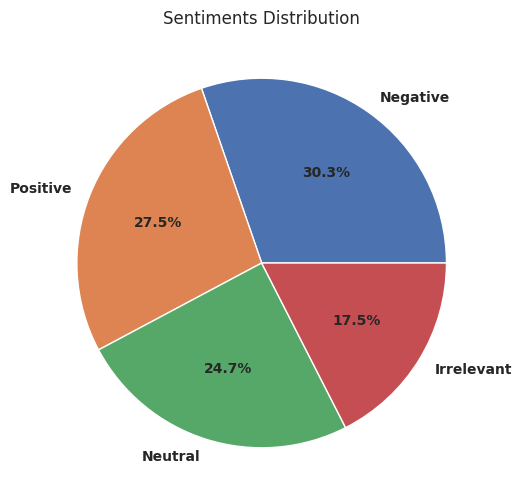
**COUNT PLOT OF SENTIMENTS:**



The dataset comprises four sentiment categories: **Positive**, **Neutral**, **Negative**, and **Irrelevant**. A count plot was generated to visualize the number of samples in each sentiment class. The results are as follows:

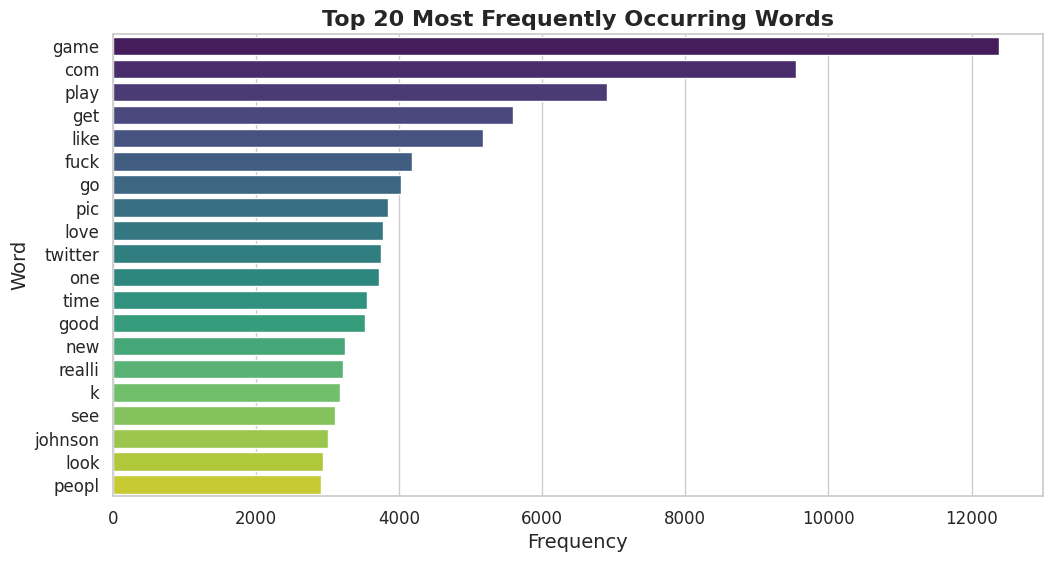
* **Negative** sentiments are the most common, with **21,698** samples.
* **Positive** sentiments follow closely, with **19,712** samples.
* **Neutral** sentiments make up **17,708** entries.
* **Irrelevant** sentiments are the least represented, totaling **12,537** entries.

**SENTIMENTS DISTRIBUTION:**



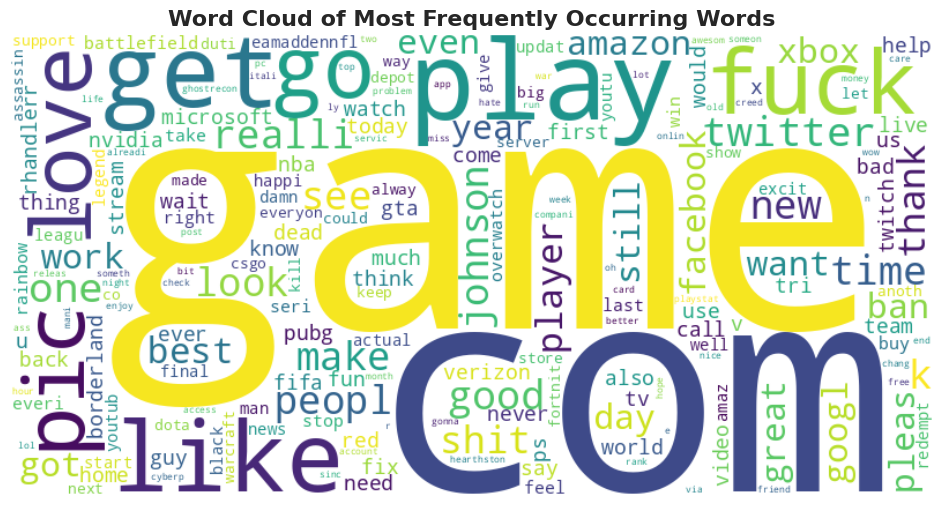
The pie chart illustrates the proportion of each sentiment category within the dataset. Among all the sentiments, **Negative** feedback constitutes the largest portion, making up approximately **30.3%** of the data. This is followed by **Positive** sentiments, which represent around **27.5%**, indicating a slightly more optimistic tone in the dataset compared to neutral statements. **Neutral** sentiments account for about **24.7%**, reflecting a significant portion of posts or reviews that convey balanced or emotionless opinions. Finally, **Irrelevant** sentiments make up the smallest share, totaling **17.5%**, suggesting that some entries are off-topic or not directly related to the main subject of analysis. This distribution is important for understanding user bias and tuning the model for balanced performance across classes.

**MOST FREQUENTLY OCCURRING WORDS:**



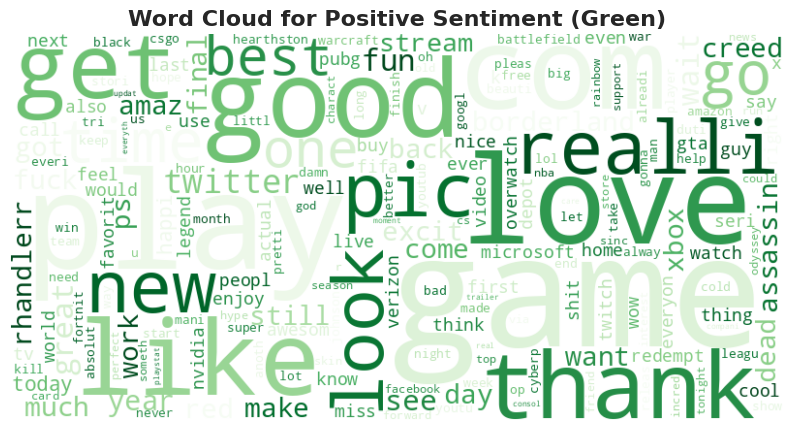
The bar plot displays the 20 most frequently occurring words in the dataset after preprocessing. The word **"game"** appears most often, with over **12,000 occurrences**, followed closely by **"com"** and **"play"**, each appearing more than **9,000** and **7,000** times respectively. Other frequently used terms include **"get"**, **"like"**, **"love"**, and **"twitter"**, indicating common themes related to gaming, online interaction, and social media. The presence of strong opinion words like **"fuck"**, **"good"**, and **"really"** reflects the emotional nature of the content. Overall, the most frequent words suggest the dataset is heavily influenced by real-time expressions, often centered around entertainment, personal reactions, and trending public figures like **"johnson"**.

**WORD COUNT OF MOST FREQUENTLY OCCURRING WORDS:**

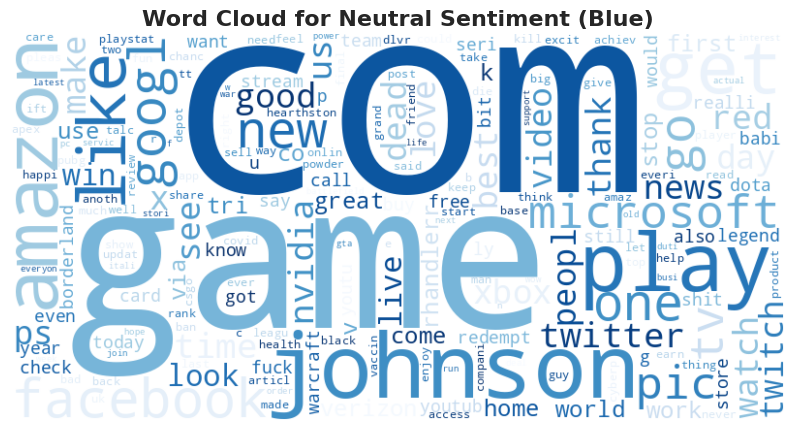


The word cloud provides a visual representation of the most frequently used words in the dataset, with larger words indicating higher frequency. Prominent terms like **"game"**, **"com"**, **"play"**, and **"get"** dominate the visualization, reinforcing the observation that much of the content revolves around gaming and online platforms. Words such as **"love"**, **"twitter"**, **"fuck"**, and **"johnson"** also appear prominently, reflecting the emotional tone and trending discussions within the dataset. The presence of terms like **"facebook"**, **"amazon"**, and **"google"** points toward frequent mentions of major tech companies. Overall, the word cloud highlights the dominant themes of gaming, social media, and public sentiment, offering quick insights into the common vocabulary and focal points of user-generated content.

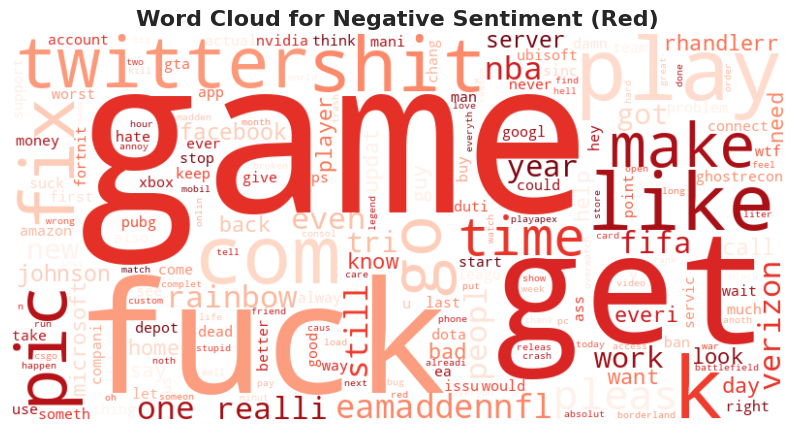
**WORD COUNT FOR POSITIVE SENTIMENT:**



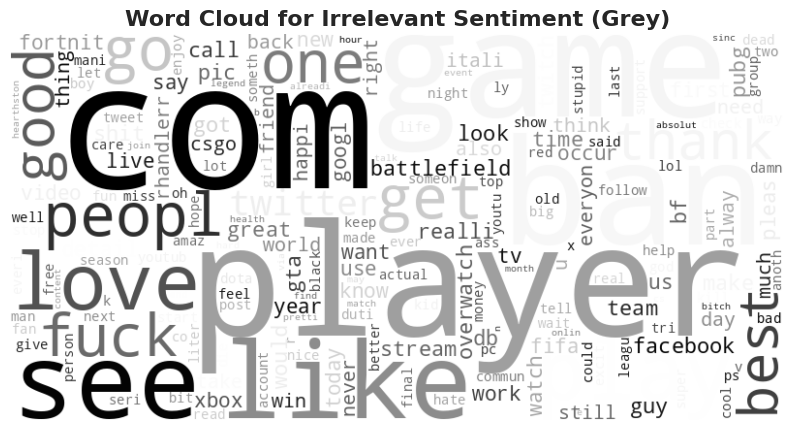
**WORD COUNT FOR NEUTRAL SENTIMENT:**



**WORD COUNT FOR NEGATIVE SENTIMENT:**



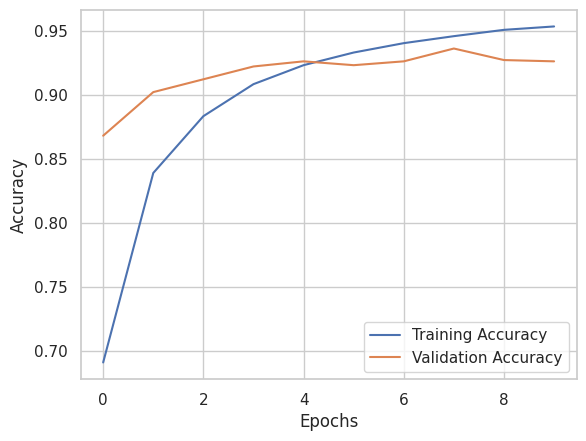
**WORD COUNT FOR IRRELEVANT SENTIMENT:**



**KERASMODEL:**

| **Layer (type)** | **Output Shape** | **Param #** |
| --- | --- | --- |
| embedding (Embedding) | ? | 0 (unbuilt) |
| lstm (LSTM) | ? | 0 (unbuilt) |
| dropout (Dropout) | ? | 0 |
| lstm\_1 (LSTM) | ? | 0 (unbuilt) |
| dropout\_1 (Dropout) | ? | 0 |
| dense (Dense) | ? | 0 (unbuilt) |
| dense\_1 (Dense | ? | 0 (unbuilt) |

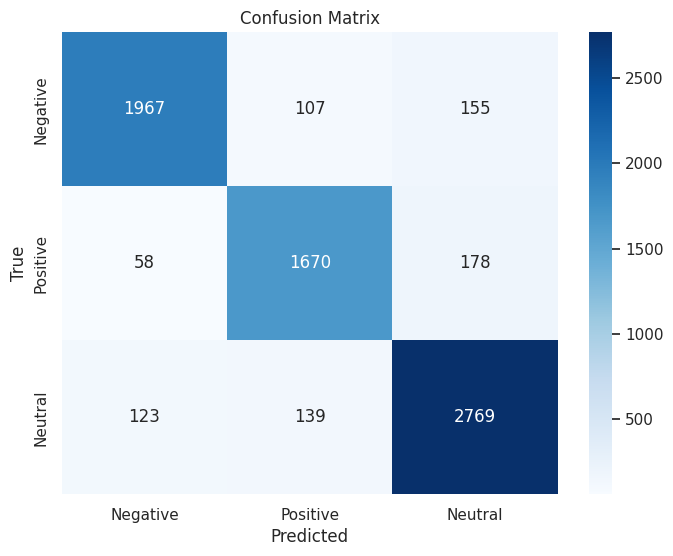
**TRAINING VS VALIDATION ACCURACY CURVE:**



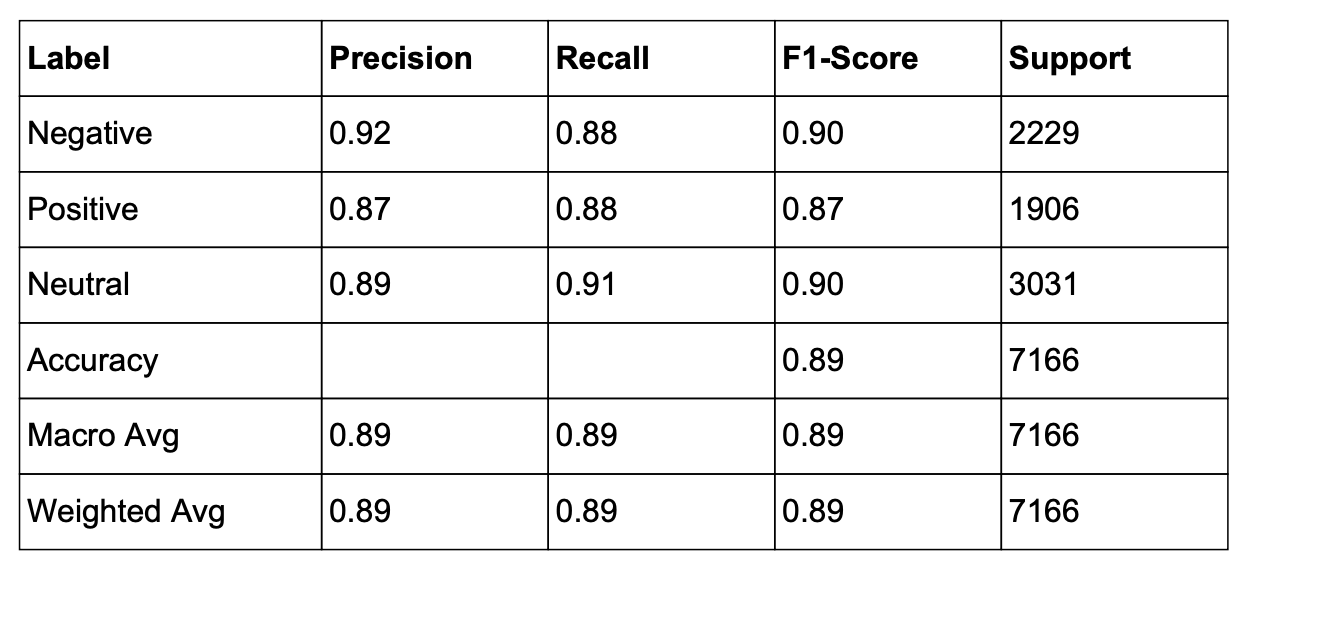
In this graph, the x-axis represents the number of epochs, which are iterations of training on the entire dataset, while the y-axis represents the accuracy — a measure of how well the model performs. Two lines are shown: one for training accuracy and the other for validation accuracy. The **training accuracy** indicates how well the model is learning from the training dataset, while the **validation accuracy** shows how well the model generalizes to unseen data.

From the beginning, the training accuracy starts lower but increases steadily as the model learns from the training data. The validation accuracy starts higher and improves initially, indicating that the model is also doing well on unseen data. However, around epoch 6 or 7, the validation accuracy stops increasing and slightly fluctuates, while the training accuracy continues to improve. This behavior suggests that the model might be **overfitting** — it performs very well on training data but starts to lose its ability to generalize on validation data. The gap between the two curves in the final epochs supports this conclusion.

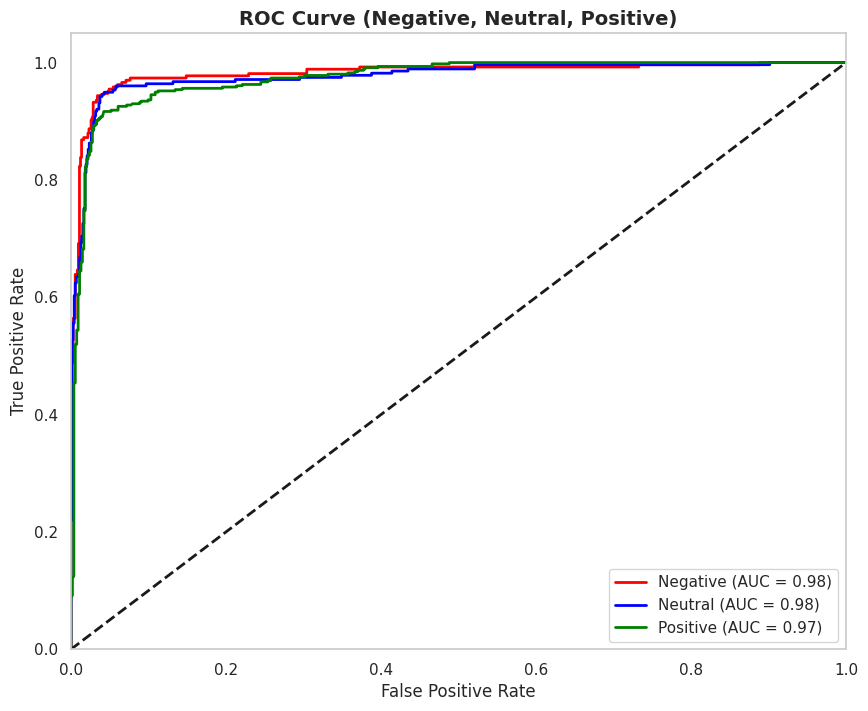
**CONFUSION MATRIX:**

****

**CLASSIFICATION REPORT:**

****

**ROC CURVE:**

****

**RESULTS:**

The code systematically processes ASL sign images through a series of steps. Essential libraries like TensorFlow, Keras, OpenCV, and scikit-learn are imported for numerical computations, image processing, and model building. Key parameters, including image size, batch size, number of epochs, and a random seed, are defined to configure the training process. To ensure reproducibility, the code sets a fixed random seed. The ASL sign image data is loaded from a specified path, and sample images are displayed to provide a visual overview of the dataset. A pandas DataFrame is generated to organize the image paths and their corresponding labels. The dataset is then divided into training, validation, and test sets, using stratified splitting to maintain class balance across the subsets. Data augmentation techniques are applied to the training set to enhance model robustness, and data generators are prepared for efficient batch processing. A custom CNN model is constructed, featuring convolutional layers, batch normalization, max pooling, and dropout. The architecture of this model is then visualized. The model is compiled with an Adam optimizer, categorical cross-entropy loss function, and relevant evaluation metrics. Callbacks are defined to save the best model and log training progress using TensorBoard. The model is trained using the training data generator, with validation performed during training. Following training, the model is evaluated on the test set, and the test accuracy is printed. The trained model is then saved for future use. Finally, the training and validation loss and accuracy curves are plotted to visualize the training process, and a confusion matrix is generated to illustrate the model's performance in classifying the different ASL signs. A detailed classification report, including precision, recall, and F1-score for each class, along with overall metrics, is also generated.

**CONCLUSION:**

In conclusion, the code successfully trains a CNN model to recognize ASL hand signs. Data augmentation proves to be a valuable technique for enhancing the model's ability to generalize to unseen data. The confusion matrix and classification report offer detailed insights into the model's performance, highlighting its strengths and weaknesses in classifying specific signs. Additionally, the training and validation loss/accuracy plots provide a clear view of the model's learning process and can be used to diagnose potential issues such as overfitting or underfitting.