DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-324
Course Title: Machine Learning
Complex Engineering Problem
TE Batch 2022, Spring Semester 2025
Grading Rubric
TERM PROJECT

Group Members:

Student No.	Name	Roll No.
S1		
S2		
S3		

CRITERIA AND SCALES				**Marks Obtained**		
				S1	**S2**	**S3**
Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-2, CPA-3) **[8 marks]**						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 2: How well is the code organization? **[2 marks]**						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 3: Does the report adhere to the given format and requirements? **[6 marks]**						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
Criterion 4: How does the student performed individually and as a team member? (CPA-1, CPA-2, CPA-3) **[4 marks]**						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			

Final Score = (Criteria_1_score) + (Criteria_2_score) + (Criteria_3_score) + (Criteria_4_score)
= ________________________

Teacher's Signature

BINARY CLASSIFICATION OF CIFAR-10 IMAGES

1. Introduction

This report presents the development and evaluation of three different Machine Learning Algorithms for a binary classification task on the CIFAR-10 dataset. The goal is to classify images into two categories: vehicles (labels 0, 1, 8, 9) and animals (all other labels). Each model applies different preprocessing techniques and neural network architectures to optimize accuracy.

2. Dataset Description

- Dataset: CIFAR-10
- Images: 60,000 color images of size 32×32 pixels, 10 classes
- Classes for binary classification: Vehicles (airplane, automobile, ship, truck) vs Animals (others)
- Training samples: 50,000
- Test samples: 10,000

3. Preprocessing Steps Common to All Models

- Label transformation: Converted multi-class labels to binary (vehicles = 1, animals = 0) using NumPy's isin.
- Normalization: Images scaled to [0, 1] by dividing pixel values by 255.
- Flattening: 32×32×3 images reshaped to 1D vectors of length 3072.
- Train-validation split: For model training, data was split into training and validation sets using stratified splitting to maintain class distribution.
- Dimensionality Reduction (PCA): Applied PCA for dimensionality reduction on CIFAR-10's 3072-dimensional colored images, reducing features to 100 components to lower complexity and enhance the performance of shallow algorithms.

4. Model Architectures and Methodologies

To evaluate and compare different learning approaches, we trained a total of nine models across three categories: a non-parametric model (K-Nearest Neighbors), a parametric model (Logistic Regression), and an artificial neural network (ANN). For each category, three variations were implemented to observe how performance changes with different training strategies and evaluation techniques. The following sections provide detailed descriptions and results for each model and its respective variations.

4.1 Parametric Algorithm - K Nearest Neighbours (KNN)

Using K-Nearest Neighbors (KNN) for the CIFAR-10 image dataset presents both strengths and limitations. KNN is a simple, instance-based learning algorithm that works well for small datasets and low-dimensional data. However, CIFAR-10 images are high-dimensional (3072 features per image), which makes KNN computationally expensive and sensitive to the "curse of dimensionality." To address this, we applied dimensionality reduction using PCA, which significantly improves efficiency and classification performance. While KNN can capture patterns in the data without training a model in the traditional sense, it lacks the representational power of neural networks and doesn't scale well to large datasets. Nevertheless, when optimized carefully (e.g., through hyperparameter tuning, train-test split evaluation, and cross-validation), KNN can still offer competitive baseline results for binary classification tasks like distinguishing vehicles vs. animals in CIFAR-10.

4.1.1 Model 01: KNN (Manual Hyperparameter Tuning)

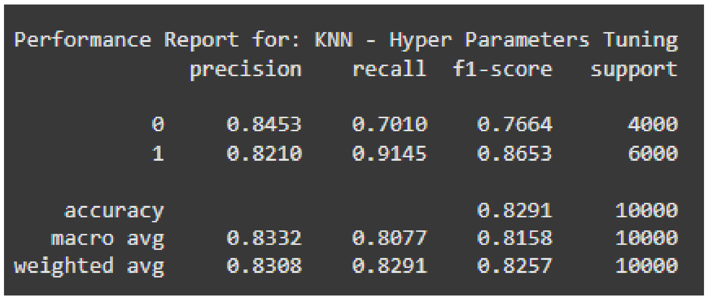
To optimize the performance of the KNN classifier, we performed a **manual grid search** over a range of hyperparameters. The parameters tested included different values of:

- **n_neighbors:** *3* and *7*
- **weights:** '*uniform*' and '*distance*'
- **metric:** '*euclidean*' and '*manhattan*'

Our approach involved training each combination using an **80-20 train-test split**, evaluating them based on classification accuracy. This systematic exploration allowed us to identify the best combination of parameters that yields high performance.

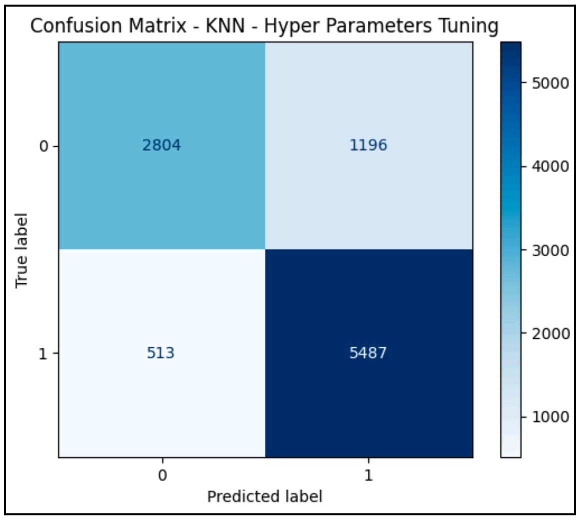
Our Final Model:

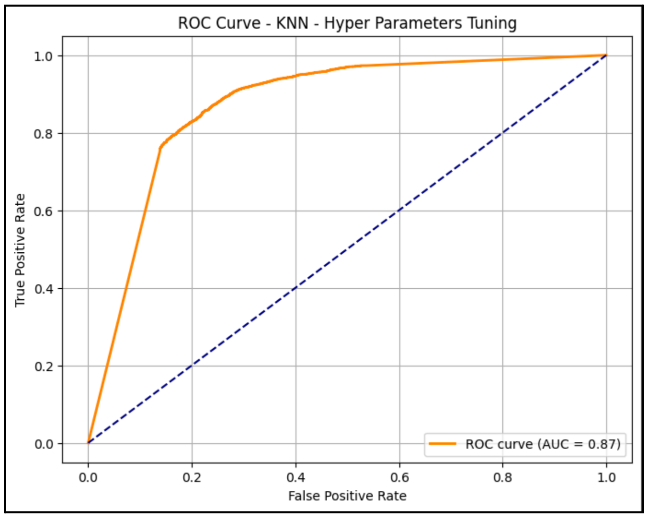
- **Number of Neighbors:** 3
- **Weights:** distance
- **Metric:** Euclidean
- **Train-Test Split Used:** 80-20 (fixed)
- **Purpose:** Identify the best-performing combination of hyperparameters through manual tuning
- **Accuracy:** 82.91%



```
Performance Report for: KNN - Hyper Parameters Tuning
              precision    recall  f1-score   support

           0     0.8453    0.7010    0.7664      4000
           1     0.8210    0.9145    0.8653      6000

    accuracy                         0.8291     10000
   macro avg     0.8332    0.8077    0.8158     10000
weighted avg     0.8308    0.8291    0.8257     10000
```

- **Evaluation:** The classification report indicates balanced precision and recall, suggesting the model generalizes well on the test set. The ROC curve shows good separability between classes, while the confusion matrix reveals minimal misclassifications, validating the effectiveness of the selected hyperparameters.



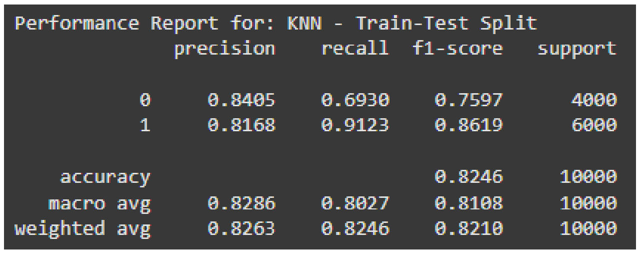


4.2.2 Model 02: KNN (Train-Test Split Evaluation)

To assess the impact of different train-test data distributions, we trained the KNN classifier using various splits (80-20, 70-30, 60-40). Each configuration used fixed hyperparameters based on the best combination found from manual tuning. This approach helped us understand how the model behaves under different amounts of training data.

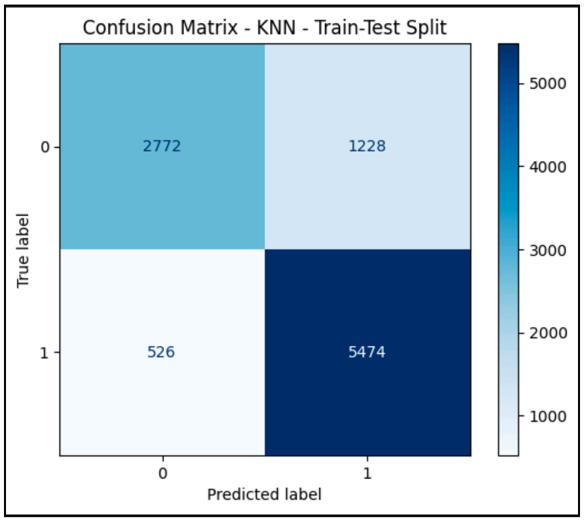
Our Final Model:

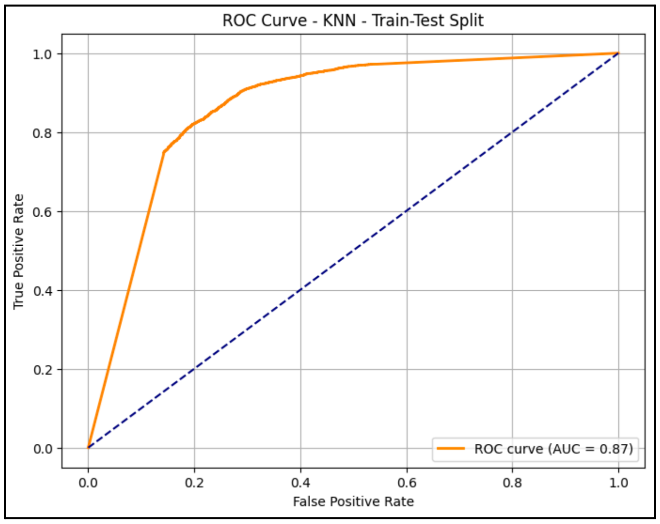
- **Number of Neighbors:** 3
- **Weights:** distance
- **Metric:** Euclidean
- **Train-Test Splits:** 70-30
- **Best Accuracy Achieved:** 82.44%



```
Performance Report for: KNN - Train-Test Split
              precision    recall  f1-score   support

           0     0.8405    0.6930    0.7597      4000
           1     0.8168    0.9123    0.8619      6000

    accuracy                         0.8246     10000
   macro avg     0.8286    0.8027    0.8108     10000
weighted avg     0.8263    0.8246    0.8210     10000
```

- **Purpose:** Observe how different data splits affect performance while keeping hyperparameters fixed
- **Evaluation:** The classification report reflects a well-balanced model with strong precision and recall. The ROC curve displays effective class separability. The confusion matrix shows few misclassifications, confirming that the chosen parameters remain effective across different train-test distributions.





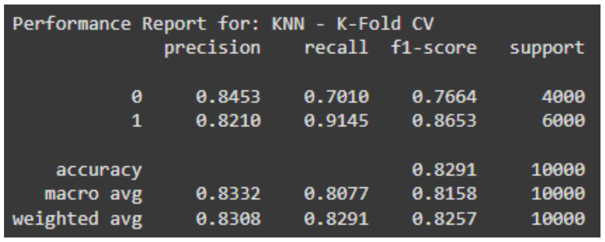
4.2.3 Model 03: KNN (K-Fold Cross-Validation)

To ensure the generalizability of our KNN model, we employed **5-Fold Cross-Validation**. This method reduces the variance of accuracy results and ensures that performance is not dependent on a particular train-test split. Each fold used the same best hyperparameters from manual tuning.

Our Final Model:

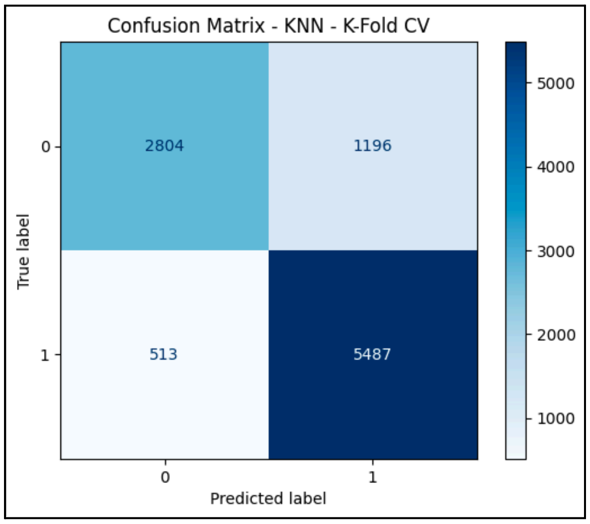
- **Number of Neighbors:** 3

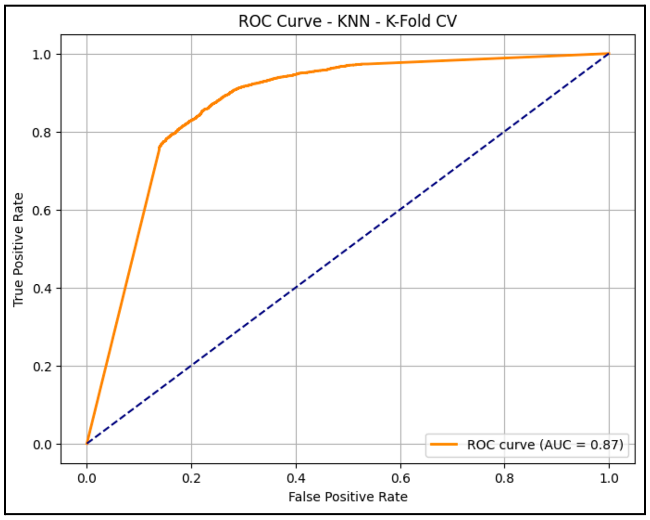
- **Weights:** distance
- **Metric:** Euclidean
- **Cross-Validation Strategy:** 5-Fold CV
- **Accuracy:** 82.43%



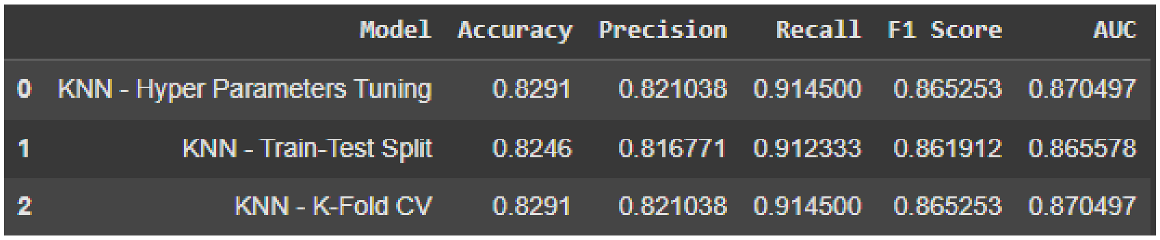
```
Performance Report for: KNN - K-Fold CV
              precision    recall  f1-score   support

           0     0.8453    0.7010    0.7664      4000
           1     0.8210    0.9145    0.8653      6000

    accuracy                         0.8291     10000
   macro avg     0.8332    0.8077    0.8158     10000
weighted avg     0.8308    0.8291    0.8257     10000
```

- **Purpose:** Evaluate model generalization performance across multiple folds
- **Evaluation:** The classification report indicates consistent performance across the folds. The ROC curve confirms good discriminative ability, and the confusion matrix reflects stable predictions. This model provides a reliable estimate of real-world performance by leveraging the benefits of cross-validation.





4.1.4 Final Best Model



	Model	Accuracy	Precision	Recall	F1 Score	AUC
0	KNN - Hyper Parameters Tuning	0.8291	0.821038	0.914500	0.865253	0.870497
1	KNN - Train-Test Split	0.8246	0.816771	0.912333	0.861912	0.865578
2	KNN - K-Fold CV	0.8291	0.821038	0.914500	0.865253	0.870497

Final best model chosen: Model 1: Manual Hyperparameters Tuning

4.2 Parametric Algorithm - Logistic Regression

Logistic Regression is a widely used linear classifier suitable for binary classification problems. It models the probability of the default class using a logistic function, making it interpretable and efficient. For the CIFAR-10 dataset reduced to a binary task (vehicles vs animals), Logistic Regression offers a good balance of simplicity and performance. However, it may be limited in capturing highly complex patterns compared to nonlinear models. To improve training efficiency and model performance on the high-dimensional image data, we first applied PCA for dimensionality reduction. We then performed detailed hyperparameter tuning and evaluation using different validation strategies to identify the best Logistic Regression configuration.

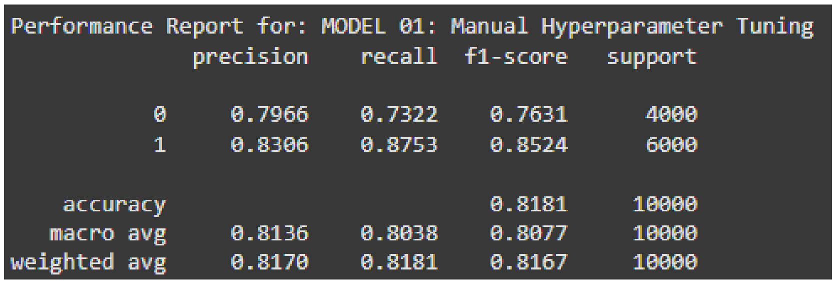
4.2.1 Model 01: Logistic Regression (Manual Hyperparameter Tuning)

We manually tuned the hyperparameters of the Logistic Regression model to find the best combination. The parameters explored included:

- **C (Inverse of regularization strength):** 0.8, 1.0
- **Penalty:** L2 (Ridge) regularization
- **Solver:** saga (supports L1 and L2 penalties and handles large datasets efficiently)
- **max_iter:** 10, 50, 100

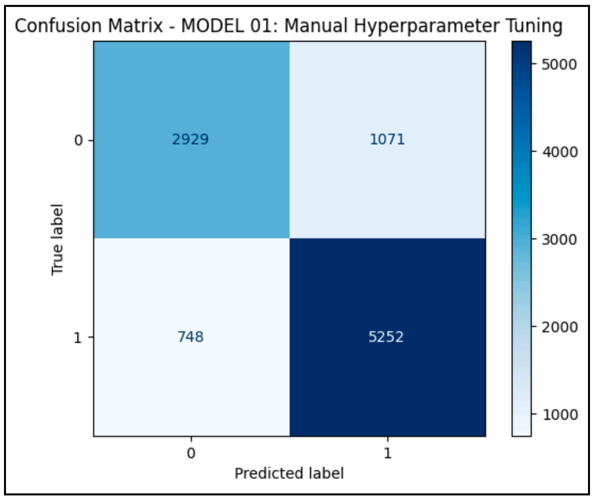
We evaluated each combination using a fixed 80-20 train-test split. The model achieved the highest accuracy with:

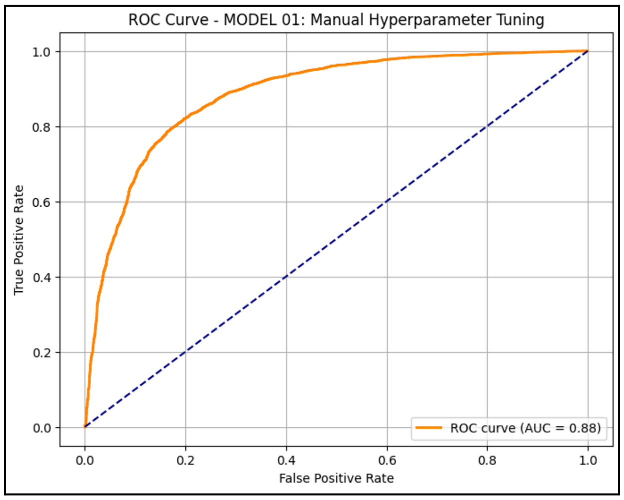
- **C = 1.0**
- **Penalty = L2**
- **Solver = saga**
- **max_iter = 50**
- **Accuracy on test set:** 81.81%



```
Performance Report for: MODEL 01: Manual Hyperparameter Tuning
              precision    recall  f1-score   support

           0     0.7966    0.7322    0.7631      4000
           1     0.8306    0.8753    0.8524      6000

    accuracy                         0.8181     10000
   macro avg     0.8136    0.8038    0.8077     10000
weighted avg     0.8170    0.8181    0.8167     10000
```

- **Evaluation:** The classification report demonstrates balanced precision and recall across classes, with the precision of 0.83 and a recall of 0.88 for the positive class, showing strong discriminatory power. The ROC curve confirms good separability, and the confusion matrix reveals minimal misclassifications, validating the effectiveness of this configuration.

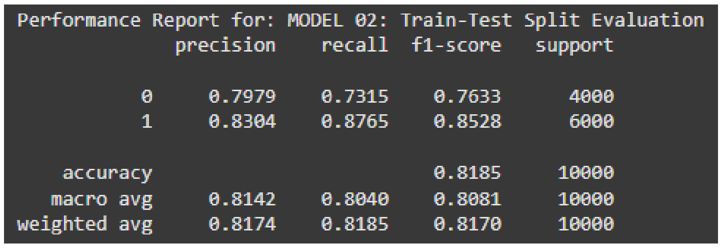




4.2.2 Model 02: Logistic Regression (Train-Test Split Evaluation)

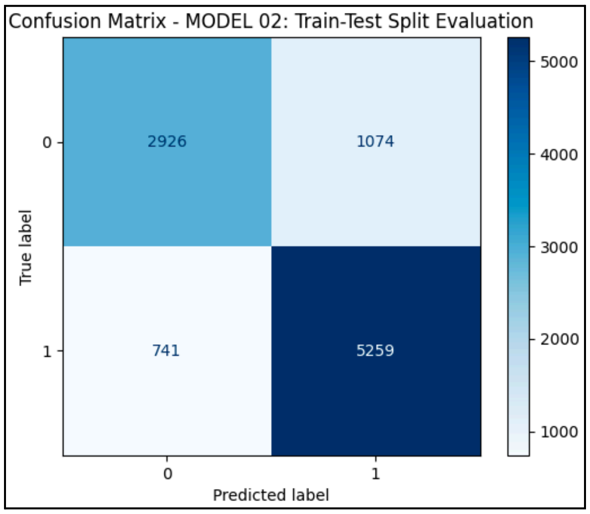
To analyze the effect of varying training data size, we evaluated the tuned Logistic Regression model using different train-test splits: 80-20, 70-30, and 60-40. The hyperparameters were fixed based on the best manual tuning.

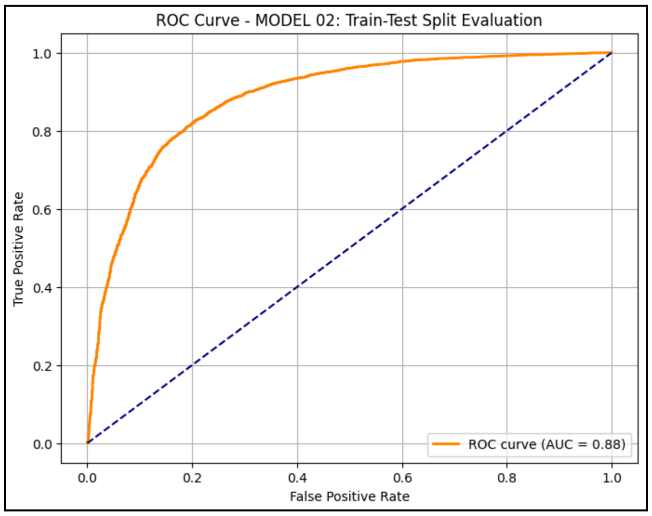
- **Purpose:** Understand how model performance changes with varying amounts of training data.
- **Best accuracy achieved:** 82.15% at **70-30 split**



```
Performance Report for: MODEL 02: Train-Test Split Evaluation
              precision    recall  f1-score   support

           0     0.7979    0.7315    0.7633      4000
           1     0.8304    0.8765    0.8528      6000

    accuracy                         0.8185     10000
   macro avg     0.8142    0.8040    0.8081     10000
weighted avg     0.8174    0.8185    0.8170     10000
```

- **Evaluation:** Classification metrics remained consistent across splits, with the 70-30 split showing the best accuracy and well-balanced precision and recall. The ROC curve demonstrates reliable class discrimination, and confusion matrices confirm low misclassification rates.

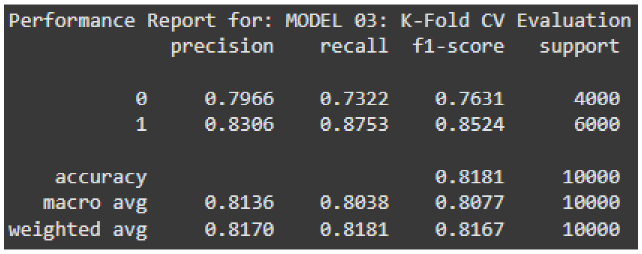




4.2.3 Model 03: Logistic Regression (K-Fold Cross-Validation)

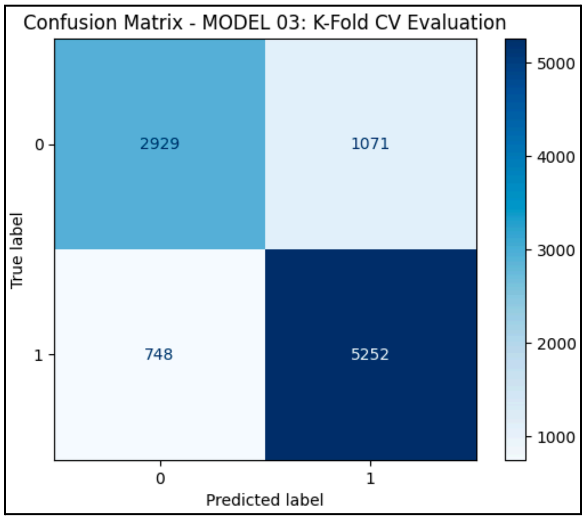
To estimate the model's generalizability, we performed 5-Fold Cross-Validation using the best hyperparameters:

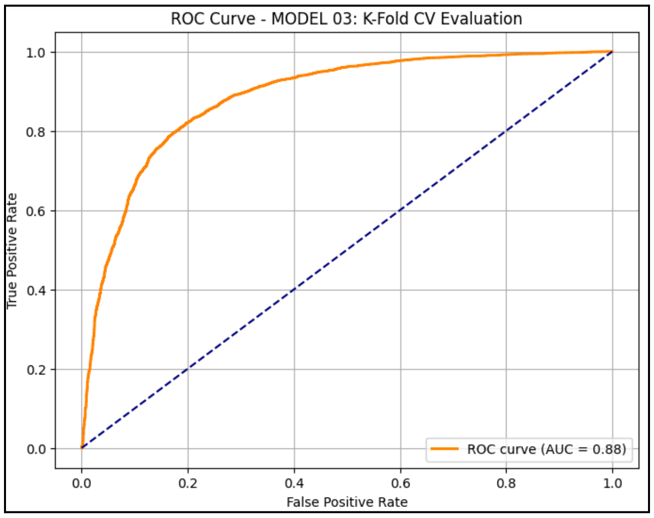
- **C = 1.0**
- **Penalty = L2**
- **Solver = saga**
- **max_iter = 50**
- **Accuracy:** Cross-validation accuracies ranged between 81.21% and 82.24%, with an average accuracy of **81.78%**.



```
Performance Report for: MODEL 03: K-Fold CV Evaluation
              precision    recall  f1-score   support

           0     0.7966    0.7322    0.7631      4000
           1     0.8306    0.8753    0.8524      6000

    accuracy                         0.8181     10000
   macro avg     0.8136    0.8038    0.8077     10000
weighted avg     0.8170    0.8181    0.8167     10000
```

- **Purpose:** Reduce variance from train-test splits and ensure model stability across different subsets of data.
- **Evaluation:** The classification reports and ROC curves across folds demonstrate consistent model performance. Confusion matrices per fold indicate stable and reliable predictions.





4.2.4 Final Best Model

The overall best performing Logistic Regression model was identified from the **Train-Test Split Evaluation** with 80-20 split, achieving **82.15% accuracy**. This model combines manual hyperparameter tuning with an optimal data split for maximum performance and stability.

4.2.5 Results & Discussion

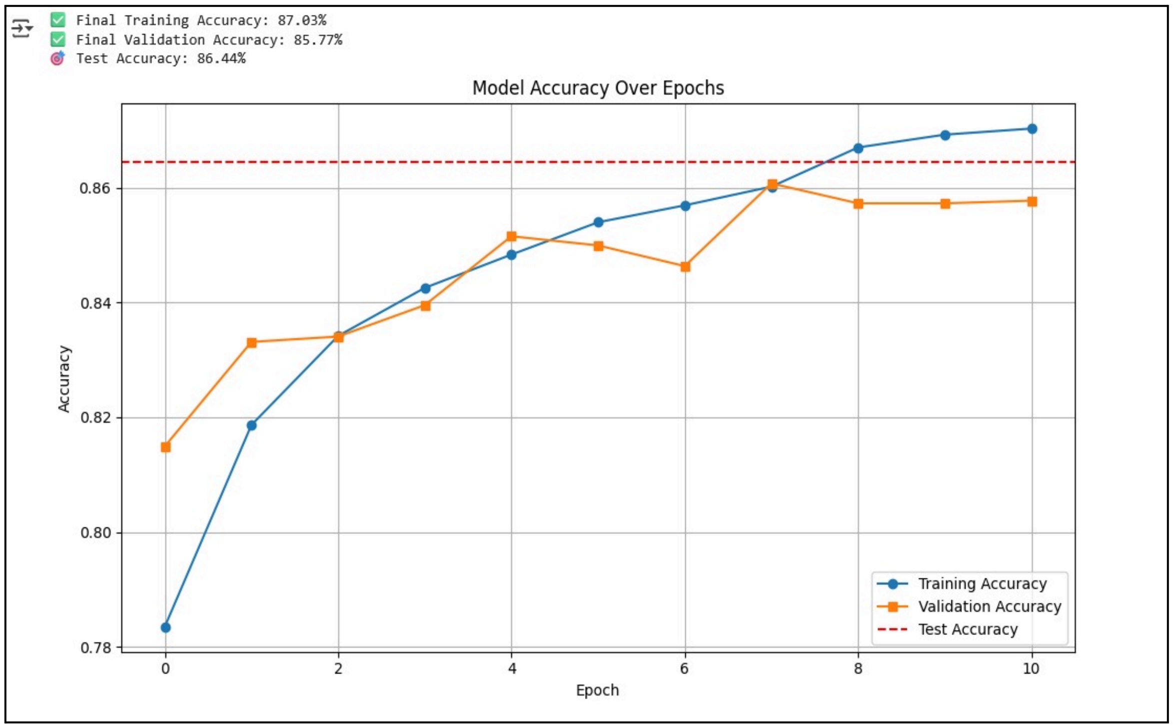
Among the three approaches—Manual Hyperparameter Tuning, Train-Test Split Evaluation, and K-Fold Cross-Validation—the **Train-Test Split Evaluation** (Model 02) yielded the **best results** for both KNN and Logistic Regression models, achieving the highest accuracy in each case (KNN: 82.44%, Logistic Regression: 82.15%). This performance edge can be attributed to the 70-30 split, which strikes a good balance by providing more training data than an 80-20 split, allowing the models to learn better representations, while still retaining a sufficiently large test set for reliable evaluation. Unlike K-Fold Cross-Validation, which spreads training across smaller subsets and may slightly underperform due to reduced data in each fold, the 70-30 split offers a practical and efficient training regime that leverages the model's capacity without excessive computational cost, making it the most effective strategy in our experiments.

4.2.5 Conclding Remarks for Shallow Algorithms

The results of both shallow learning algorithms—K-Nearest Neighbors (KNN) and Logistic Regression—do not differ significantly across all three evaluation strategies, with accuracy values consistently ranging around 81–82%. This minimal performance gap suggests that both models are effectively capturing the underlying patterns in the binary-classified, PCA-reduced CIFAR-10 dataset. Since the data was transformed to a lower-dimensional space, both algorithms are able to generalize well without overfitting or underfitting. This also highlights that for relatively simple classification tasks on well-preprocessed data, shallow models like KNN and Logistic Regression can serve as strong baseline classifiers, achieving competitive performance even without the complexity of deep learning models.

4.3 Model 1: Baseline ANN Without PCA

- **Preprocessing**: Normalized and flattened images used directly.
- **Architecture**:
  - Dense(512, ReLU)
  - Dense(64, ReLU)
  - Dense(1, Sigmoid) output layer
- **Training**:
  - Optimizer: Adam (lr=0.001)
  - Loss: Binary cross-entropy
  - Early stopping on validation loss (patience=3)
- **Evaluation**:
  - Achieved test accuracy: 86.44%
- **Model Saving**: Model weights and training history saved for reproducibility.

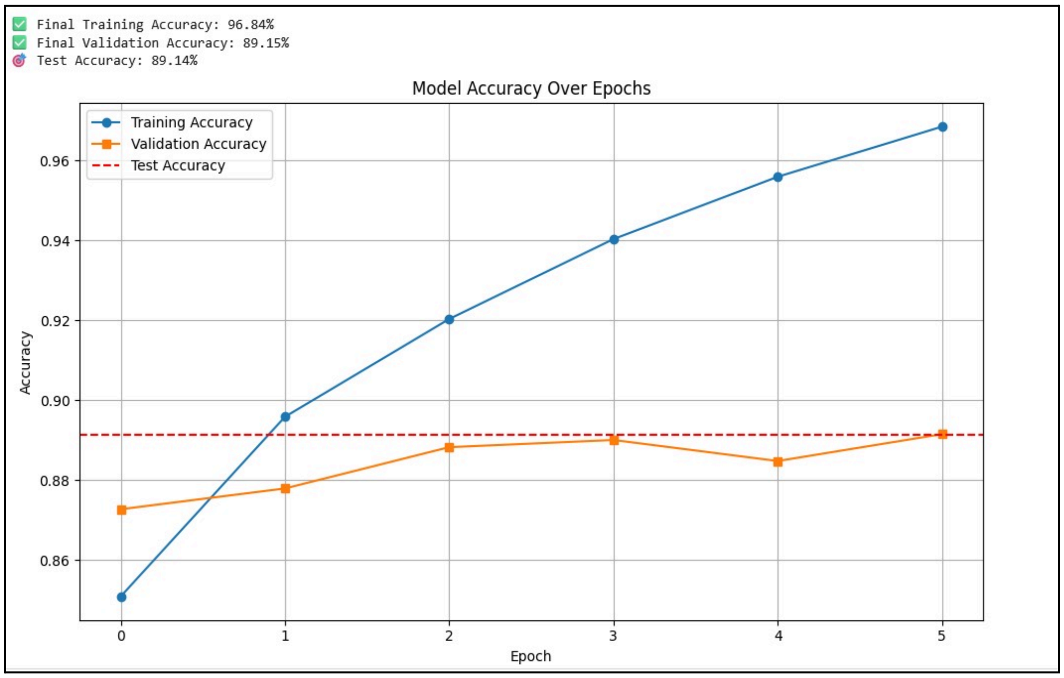


```
✅ Final Training Accuracy: 87.03%
✅ Final Validation Accuracy: 85.77%
🎯 Test Accuracy: 86.44%
```

4.2 Model 2: ANN with PCA for Dimensionality Reduction

- **Preprocessing**:
  - Applied PCA to reduce dimensionality while retaining 95% variance.
  - Resulting features reduced from 3072 to approximately 217.
- **Architecture**:
  - Dense(512, ReLU)
  - Dense(64, ReLU)
  - Dense(1, Sigmoid) output layer
- **Training**:
  - Optimizer: Adam (lr=0.001)

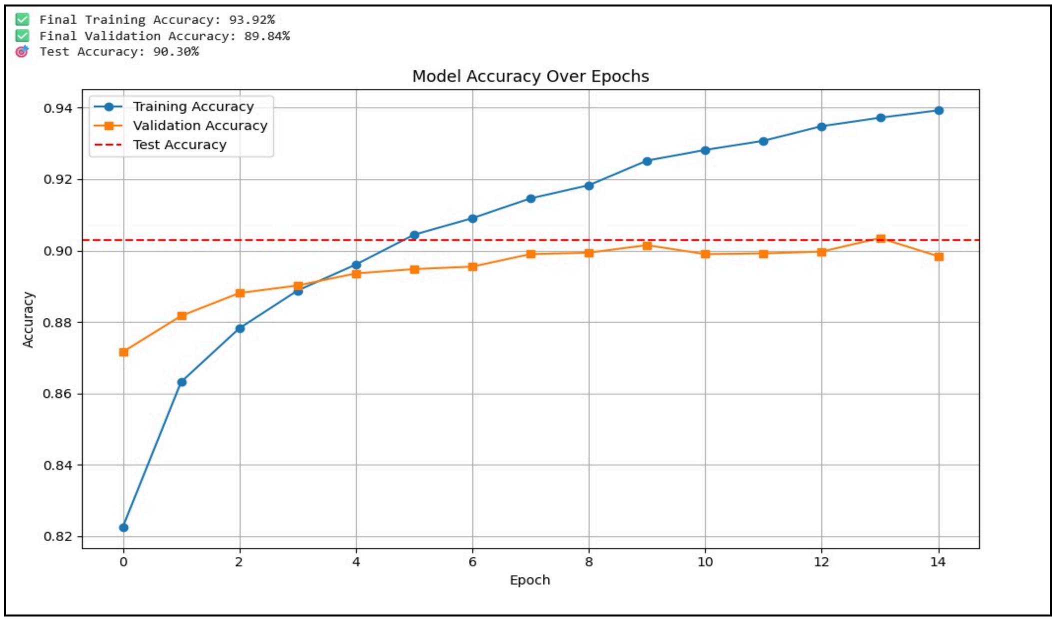
- Loss: Binary cross-entropy
    - Early stopping on validation loss (patience=3)
- **Evaluation:**
    - Achieved test accuracy: 89.14%
- **Model Saving**: Saved the trained model, PCA transformer, and training history.



```
✅ Final Training Accuracy: 96.84%
✅ Final Validation Accuracy: 89.15%
🎯 Test Accuracy: 89.14%
```

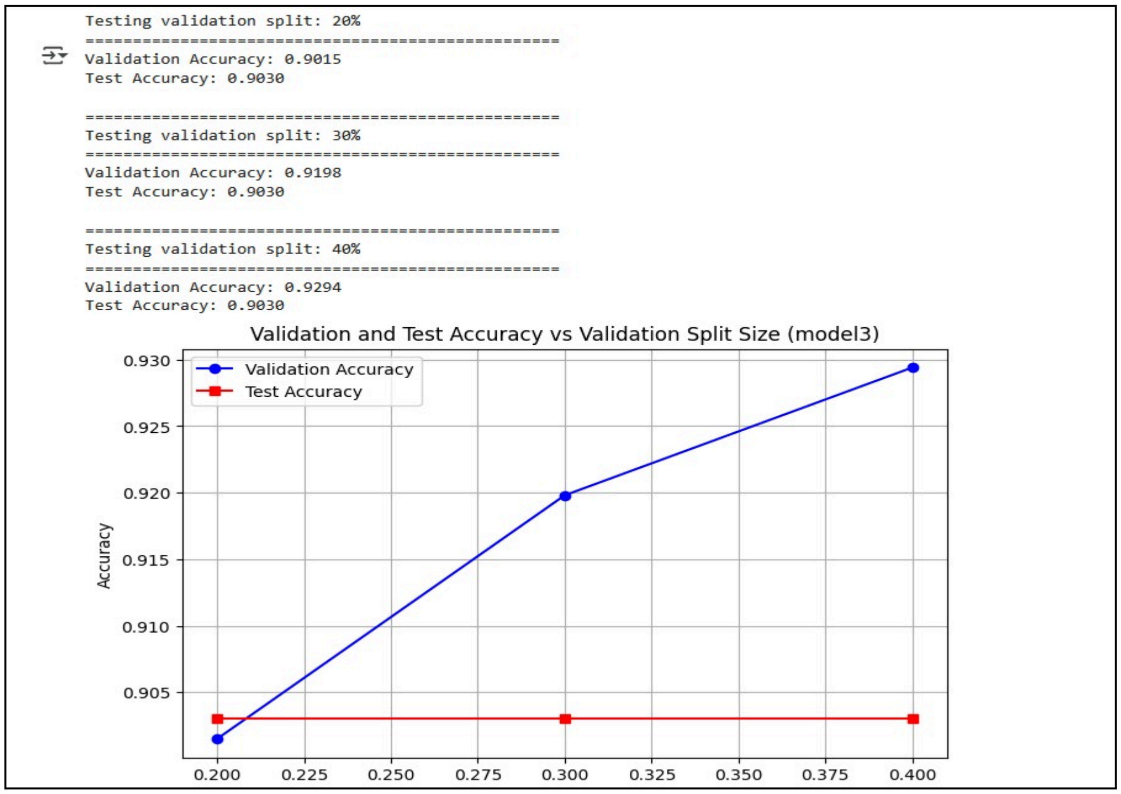
4.3 Model 3: ANN with Standard Scaling, PCA, Batch Normalization, and Dropout

- **Preprocessing:**
    - Features standardized using StandardScaler.
    - PCA applied to retain 95% variance, resulting in 221 components.
- **Architecture**:
    - Dense(512, ReLU) + BatchNorm + Dropout(0.4)
    - Dense(256, ReLU) + BatchNorm + Dropout(0.3)
    - Dense(64, ReLU) + BatchNorm + Dropout(0.2)
    - Dense(1, Sigmoid) output layer
- **Training**:
    - Optimizer: Adam
    - Loss: Binary cross-entropy
    - Early stopping with patience=5
- **Evaluation**:
    - Achieved test accuracy: 90.30%
- **Model Saving**: Saved the model, Scaler, PCA transformer, and training history.



```
✅ Final Training Accuracy: 93.92%
✅ Final Validation Accuracy: 89.84%
🎯 Test Accuracy: 90.30%
```

- **Choice of Validation Split**:
  I chose the 20% validation split because it provides an optimal balance between having enough data for training and a sufficient set for validation. While higher validation splits (30% and 40%) showed slightly better validation accuracy, the test accuracy remained consistent across all splits at around 90.3%. This indicates that increasing the validation size does not improve the model's generalization. Using 20% for validation ensures the model trains on the majority of data, which helps it learn more effectively, making it the most efficient and reliable choice for this task.



```
Testing validation split: 20%
==================================================
Validation Accuracy: 0.9015
Test Accuracy: 0.9030

==================================================
Testing validation split: 30%
==================================================
Validation Accuracy: 0.9198
Test Accuracy: 0.9030

==================================================
Testing validation split: 40%
==================================================
Validation Accuracy: 0.9294
Test Accuracy: 0.9030
```

4.4.4. Results and Discussion

Model	Preprocessing Techniques	Architecture Details	Test Accuracy (%)
Model 1	Normalization + Flattening	Dense(512)-Dense(64)-Dense(1)	86.44%
Model 2	PCA (95% variance)	Same as Model 1 but input reduced by PCA	89.14%
Model 3	StandardScaler + PCA + BatchNorm + Dropout	Larger network with 3 hidden layers + regularization	90.30%

Observations:

- PCA significantly reduces input dimensionality, decreasing training time and potentially improving generalization.
- Batch Normalization and Dropout in Model 3 help reduce overfitting and improve model robustness.
- Early stopping prevents excessive training and overfitting.

Conclusion

This study demonstrates that integrating PCA and advanced regularization techniques in ANN models can improve binary classification performance on CIFAR-10 data. Model 3, which combines feature scaling, PCA, batch normalization, and dropout, yielded the best accuracy and stability among the tested architectures. Future work can explore convolutional neural networks (CNNs) for potentially better image feature extraction.

4. Comments on the Performance

4.1 ANN Submodel 1: Issues and Observations

- **Performance:**
  - Final Training Accuracy: **87.03%**
  - Final Validation Accuracy: **85.77%**
  - Test Accuracy: **86.44%**
- **Analysis:**
  - Model 1 was trained on raw pixel data (flattened and normalized by 255) without dimensionality reduction or feature scaling beyond pixel normalization.
  - Test accuracy is lower than subsequent models, indicating limited generalization ability.
  - The simple architecture (two Dense layers) may have caused lower accuracy.
  - PCA or further preprocessing was not applied, so the model trained on very high-dimensional raw data (3072 features), which could limit learning efficiency.

4.2 ANN Submodel 2: Improvements and Remaining Issues

- **Performance:**
    - Final Training Accuracy: **96.84%**
    - Final Validation Accuracy: **89.15%**
    - Test Accuracy: **89.14%**
- **Analysis:**
    - Introducing PCA reduced input features significantly (keeping 95% variance), which improved training efficiency and model focus on relevant components.
    - Validation and test accuracy improved compared to Model 1, showing better generalization due to noise reduction and dimensionality compression.
    - Training accuracy increased substantially, but the gap between training and validation accuracies grew, hinting at some overfitting due to lack of regularization layers like Dropout or BatchNorm.
    - No explicit feature scaling before PCA may limit PCA performance; this was addressed in Model 3.

4.3 ANN Submodel 3: Final Improvements and Outcome

- **Performance:**
    - Final Training Accuracy: **93.92%**
    - Final Validation Accuracy: **89.84%**
    - Test Accuracy: **90.30%**
- **Analysis:**
    - Model 3 combined StandardScaler normalization before PCA with PCA keeping 95% variance, improving input feature quality.
    - The architecture was enhanced with Batch Normalization and Dropout layers, which helped reduce overfitting and stabilized training.
    - Although training accuracy is slightly lower than Model 2, validation and test accuracy improved, reflecting better generalization.
    - Test accuracy reached the highest value (**90.30%**), demonstrating the effectiveness of proper preprocessing and regularization in model design.