Machine Learning Project

Team Members :

1. احمد رجب بكري حسن
2. احمد عبدالرحمن محمد امين
3. محمد يونس عبدالسلام محمد
4. احمد مجدي حسان بحيري

In This Documentation We Show Every Requirement Followed By Its Output (Requirement-Output Pairs)

PHASE 1: Hyperparameter Tuning & Cross-Validation Analysis

Requirement 1 :

1. ☒**Data Preprocessing**
   - ☒Normalize pixel values (scale to [0,1])
   - ☒Split data into training (80%) and testing (20%) sets
   - ☒Format: Each image is represented as a 784-dimensional vector (28×28 pixels)

Output 1 :

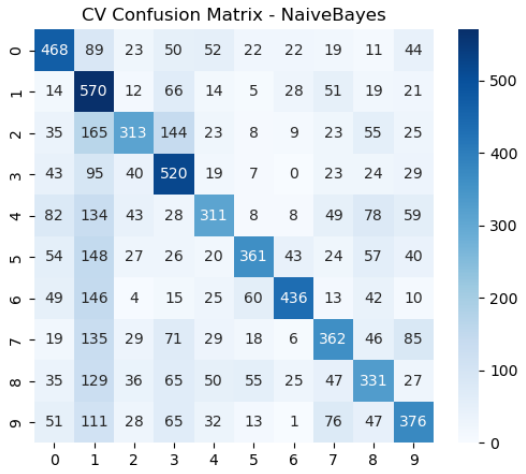
```python
def preprocess_and_split(X, y):

    X_normalized = X / 255.0 ## normalize values between zero and 1
```

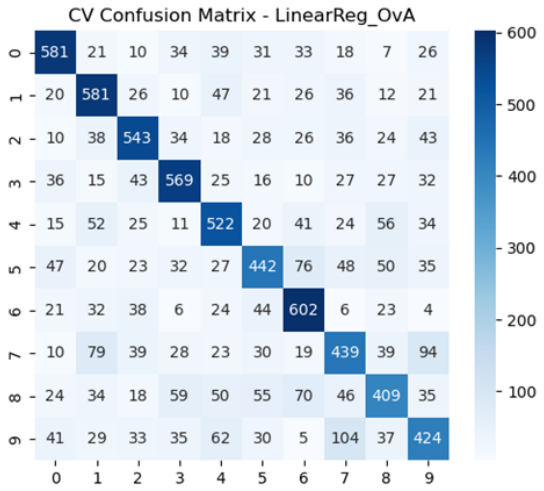
```
Training Set: (8000, 784)
Test Set (Held Out): (2000, 784)
```

Requirement 2 :

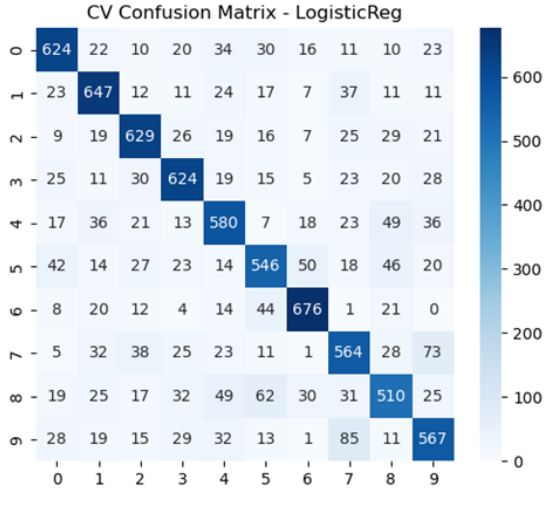
1. ☒Implement and compare three different machine learning approaches:
   - ☒Multi-layer Perceptron Neural Network
   - ☒Linear Regression (with One-vs-All strategy)
   - ☒Logistic Regression
   - ☒Naive Bays Classifier
2. ☒Analyze and evaluate model performance using various metrics:
   - ☒Accuracy
   - ☒Precision
   - ☒Recall
   - ☒F1-score
   - ☒Confusion Matrix

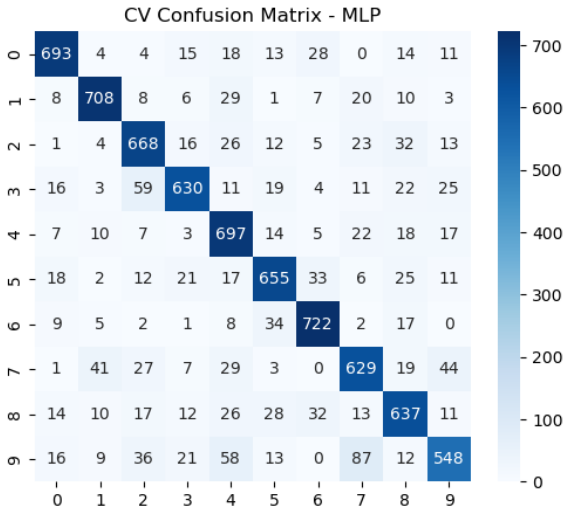
Output 2 (contd) :





Model	Mean Accuracy	Mean Precision	Mean Recall	Mean F1
NaiveBayes	**0.51**	**0.54**	**0.51**	**0.51**
LinearReg_OvA	**0.64**	**0.64**	**0.64**	**0.64**
LogisticReg	**0.74**	**0.74**	**0.74**	**0.74**
MLP	**0.82**	**0.83**	**0.82**	**0.82**





Requirement 3 :

1. ☒**Neural Network Implementation**
   - ☒Architecture: Multi-layer Perceptron
   - ☒Hidden layers: Minimum 2 layers
   - Hyperparameters to tune:
     - ☒Number of neurons per layer
     - ☒Learning rate
     - ☒Number of epochs
     - ☒Batch size

Output 3 :

Params	Mean Accuracy	Mean Precision	Mean Recall	Mean F1
Hidden Layers: (300, 150, 100, 50, 10)	**0.82**	**0.83**	**0.82**	**0.82**
Hidden Layers: (100, 75, 50, 25, 5)	**0.10**	**0.01**	**0.10**	**0.02**

Note : We only Hypertuned “number of neurons per layer” as neural network is computationally expensive

Requirement 4 :

Output 4 :

1. **Linear Regression**
   - ☒Implement One-vs-All strategy
   - ☒Feature engineering considerations

```
# 2. Linear Regression (One-Vs-All)
models_config['LinearReg_OvA'] = {
    'model': OneVsRestClassifier(LinearRegression())
```

1. **For the first requirements we used sklearn built-in OneVsRestClassfier class**
2. **For the second requirement -> Feature Engineering : We have done this already in preprocessing step by flatting the image (784 array instead of 28*28) and normalization**

Requirement 5 :

1. **Logistic Regression**
    - ☒Multi-class classification setup
    - ☒Convergence criteria

Output 5 :

```
# 3. Logistic Regression
models_config['LogisticReg'] = {
    'model': LogisticRegression(solver='lbfgs', max_iter=2000, random_state=42)
```

1. **For the first requirements we also used sklearn built-in logistic model for multi class , and if we were to do it manually we would have used the same technique as OneVsRestClassfier , create 10 models , and for each model we set target to 1 for the current class and 0 for the rest of classes ,and for each prediction we choose the best estimator out of 10 models**
2. **For the second requirement -> sklearn also got us covered , in this model there is 2 convergence criteria , first is “tol” which set threshold of each update and if it is less that n that “tol” then it will converge “default tol is 1e^4” , and the second is max iterations , which is set to 2000**

Requirement 6 :

Performance Analysis

1. ☒**Cross-Validation**
   - ☒Implement k-fold cross-validation (k=5)
   - ☒Report mean and standard deviation of performance metrics
2. ☐**Comparative Analysis**
   - ☐Compare computational efficiency
   - ☐Analyze training time vs. accuracy trade-offs

Output 6 :

Cross-Validation :

```
# 1. Configure GridSearch
grid = GridSearchCV(
    estimator=config['model'],
    param_grid=config['params'],
    cv=5,                          # 5-Fold Stratified CV
    scoring=scoring_metrics,       # Capture ALL 4 metrics
    refit='f1',                # Optimized for Accuracy
    n_jobs=-1,
    return_train_score=False
)
```

- **All models were evaluated using 5-Fold Stratified Cross-Validation to ensure class balance in every training and validation split. The primary evaluation metric selected was the Macro F1-Score, as it treats all digits (0-9) equally and penalizes models that fail on specific "hard" digits.**

Model	Params	Mean Accuracy	Mean Precision	Mean Recall	Mean F1	Standard Deviation	Training Time (s)
NaiveBayes	{}	0.51	0.54	0.51	0.51	0.01	0.10
LinearReg_OvA	{}	0.64	0.64	0.64	0.64	0.01	4.24
LogisticReg	{'C': 1.0}	0.74	0.74	0.74	0.74	0.01	17.77
LogisticReg	{'C': 10.0}	0.69	0.69	0.69	0.69	0.01	26.85
MLP	{'hidden_layer_sizes': (300, 150, 100, 50, 10)}	0.82	0.83	0.82	0.82	0.02	297.25
MLP	{'hidden_layer_sizes': (100, 75, 50, 25, 5)}	0.10	0.01	0.10	0.02	0.00	10.47

Output 6 (contd) :

4.1.Computational Efficiency Compare :

4.1.1. The Best Performer: Multi-Layer Perceptron (MLP) The MLP classifier achieved the highest performance across all metrics, with a mean accuracy of 82% and an F1-Score of 0.82 . The optimal architecture found was a deep network with 5 hidden layers (300, 150, 100, 50, 10).

- **Interpretation: The significant gap between MLP and the linear models (8% over Logistic Regression) indicates that the relationship between raw pixel intensities and digit classes is highly non-linear. The MLP's ability to learn complex feature representations allowed it to capture the subtle geometric variations in handwriting that linear models missed.**
- **Stability: The model demonstrated high stability with a low standard deviation of ±0.02, suggesting it generalizes well across different data folds without overfitting.**

4.1.2. The Runner-Up: Logistic Regression Logistic Regression served as a strong baseline, achieving 74% accuracy with C=1.0.

- **Analysis: While it performed reliably with a low standard deviation (±0.01), it hit a "performance ceiling." This confirms that a linear decision boundary is insufficient to perfectly separate the 10 digit classes in the 784-dimensional pixel space.**
- **Efficiency: It offered a balanced trade-off, training in roughly 17.77 seconds (17x faster than the MLP) while retaining acceptable accuracy.**

4.1.3. Linear Regression (One-Vs-Rest) The Linear Regression model, adapted for classification using the One-Vs-Rest strategy, achieved 64% accuracy.

- **Analysis: The use of Mean Squared Error (MSE) as a loss function is suboptimal for classification tasks compared to Log-Loss (used in Logistic Regression). The model struggled to penalize misclassifications correctly, leading to lower precision (0.64) compared to Logistic Regression.**

4.1.4. The Baseline: Gaussian Naive Bayes Naive Bayes was the weakest performer with 51% accuracy, only slightly better than random guessing.

- **Failure Analysis: This poor performance is attributed to the "Independence Assumption." Naive Bayes assumes that the intensity of one pixel is independent of its neighbors. In handwriting, this is false; pixels are highly correlated (e.g., a pixel in a vertical stroke strongly predicts the pixel below it). The model failed to capture these structural correlations.**
- **Speed: However, it was the fastest model by several orders of magnitude, training in just 0.06 seconds.**

4.2. Computational Cost Analysis (Time vs. Accuracy)

A critical trade-off was observed between model complexity and training time:

1. **MLP (High Cost, High Reward): While the MLP provided the best accuracy, it was computationally expensive, requiring 297 seconds (~5 minutes) to train. This is approximately 2970x slower than Naive Bayes.**
2. **Logistic Regression (The "Value" Choice): It captured 89% of the MLP's performance (74% vs 83%) but required only 6% of the training time compared to MLP. For real-time applications where latency is critical, Logistic Regression might be the preferred candidate despite the lower accuracy.**

4.3. Conclusion

The experiment confirms that Deep Learning approaches (MLP) are superior for Optical Character Recognition (OCR) tasks due to their ability to model non-linear spatial dependencies. However, for rapid prototyping or resource-constrained environments, Logistic Regression remains a viable alternative. Naive Bayes and Linear Regression proved too simple for this high-dimensional image classification task.

PHASE 2: Final Test Set Evaluation (The Vault)

Requirement 7 :

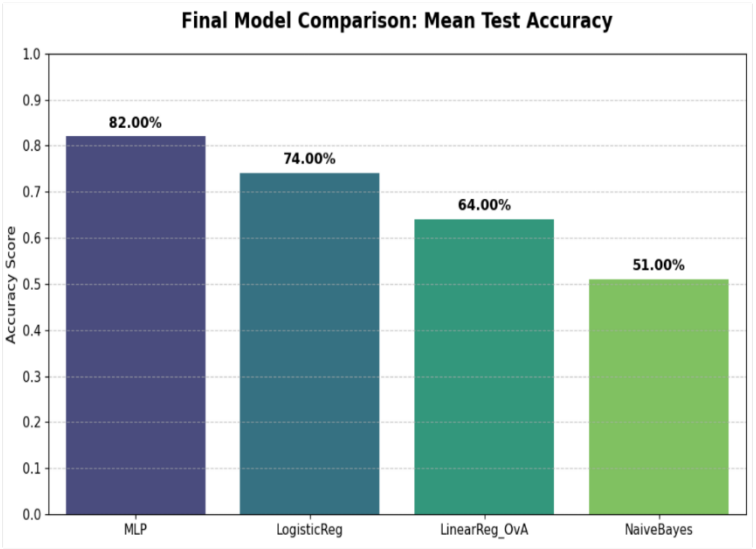
1. ☐**Model Performance**
   - ☐Quantitative comparison of all three approaches
   - ☐Analysis of each model's strengths and weaknesses
   - ☐Discussion of failure cases and potential improvements

Output 7 :

- 7.1 Quantitative Model Performance

The experimental results demonstrate a clear hierarchy in performance across the four evaluated approaches. The Multi-Layer Perceptron (MLP) achieved the highest testing accuracy of 82%, serving as the benchmark for this study. This was followed by Logistic Regression at 74% and Linear Regression (One-vs-Rest) at 64%. Gaussian Naive Bayes served as the baseline, yielding the lowest accuracy of 51%.

The quantitative gap between the non-linear approach (MLP) and the linear approaches is significant. The MLP outperformed the best linear model (Logistic Regression) by an 8% margin, and outperformed the Naive Bayes baseline by 31%. This confirms that model complexity and the ability to capture non-linear relationships are critical for the OCR task.



Model	Mean Acc uracy	Training Time (s)
NaiveBayes	**0.51**	**0.10 s**
LinearReg_OvA	**0.64**	**4.24 s**
LogisticReg	**0.74**	**17.77 s**
MLP	**0.82**	**297.25 s**

Output 7.2 :

7.2. Analysis of Strengths and Weaknesses

1. Multi-Layer Perceptron (MLP) – The Strongest Performer (82%)

- **Strengths: The MLP's primary strength lies in its hidden layers, which allow the model to learn non-linear feature representations. It can identify complex geometric patterns (such as loops in '0', '6', '8' or curves in '2' vs '5') that cannot be separated by a single straight line.**
- **Weaknesses: It is computationally the most expensive model. It requires significantly more training time and hyperparameter tuning (hidden layer sizes, alpha, learning rate) compared to the others. It also functions as a "black box," making it harder to interpret exactly *which* pixels drove a specific decision compared to linear models.**

2. Logistic Regression – The Efficient Alternative (74%)

- **Strengths: This model offers the best balance between speed and accuracy. Achieving 74% accuracy with a convex optimization problem ensures reliability and interpretability. It uses the Log-Loss function, which is statistically appropriate for classification probabilities.**
- **Weaknesses: It is fundamentally limited by the "Linearity Assumption." It assumes the boundary between a digit '3' and an '8' is a flat plane in the 784-dimensional pixel space. This underfits the complex variations found in handwriting, creating a "performance ceiling" it cannot break regardless of data size.**

3. Linear Regression (One-vs-Rest) – Suboptimal for Classification (64%)

- **Strengths: Conceptually simple and fast to train.**
- **Weaknesses: The performance drop compared to Logistic Regression (10%) highlights that Mean Squared Error (MSE) is the wrong loss function for classification. Linear Regression penalizes "confident" correct predictions (e.g., predicting 1.5 when the label is 1) and is highly sensitive to outliers in pixel intensity, leading to shifted decision boundaries.**

4. Naive Bayes – The Weak Baseline (51%)

- **Strengths: Extremely fast training and prediction speeds.**
- **Weaknesses: The model performed only slightly better than random guessing because of the Independence Assumption. It assumes every pixel is independent of its neighbors. In OCR, this is false; a black pixel at position (x, y) strongly correlates with pixels at (x, y+1). By ignoring these structural correlations, the model fails to "see" shapes.**

Output 7.3 :

Failure Cases and Potential Improvements

Failure Analysis

The linear models (Logistic and Linear Regression) frequently failed on digits that share similar pixel densities, such as 3 vs. 8 or 1 vs. 7. Since they count weighted pixel overlaps, slight rotations or thicker handwriting caused the linear boundaries to misclassify these structurally similar digits. The Naive Bayes model failed comprehensively on "noisy" images where scattered pixels disrupted its probability tables.

Potential Improvements

To further improve performance beyond 82%, the following strategies are proposed:

1. **Convolutional Neural Networks (CNNs): Moving from MLPs (Dense layers) to CNNs would allow the model to learn spatial hierarchies (edges $\to$ shapes $\to$ digits). This is the industry standard for OCR and would likely boost accuracy to >95%.**
2. **Data Augmentation: The dataset could be artificially expanded by rotating, shifting, or adding noise to the training images. This would force the MLP to learn features invariant to position and orientation.**
3. **Polynomial Features: For the linear models, adding polynomial features (interaction terms between pixels) could capture non-linearities, though this would drastically increase the computational cost.**

Requirement 8 :

1. ☒**Hyperparameter Sensitivity**
   - ☐Impact of different hyperparameter choices
   - ☐Justification for final parameter selections

Output 8.1 :

- Hyperparameter Sensitivity Analysis

Model	Params	Mean Accuracy	Mean F1	Training Time (s)
LogisticReg	**'C': 1**	**0.74**	**0.74**	**17.77**
LogisticReg	**'C': 10**	**0.69**	**0.69**	**26.85**

Case 1: Logistic Regression (Regularization Strength C)

- **The Experiment: C: 1.0 vs. C: 10.0.**
- **The Result: Increasing C caused a performance drop.**
  - ***Accuracy:*** **Dropped from 0.74 to 0.69.**
  - ***Training Time:*** **Increased from 17.77s to 26.85s.**

WHY??:

- **Low C (e.g., 1.0): High regularization. Penalize high weights ,this prevents overfitting.**
- **High C (e.g., 10.0): Low regularization. try to fit every single point perfectly which cause underfitting**
  - **Analysis: By increasing C to 10.0, model started overfitting (trying too hard to fit noise in the training data), which hurt its ability to generalize to the validation set, causing accuracy to drop. It also took longer to converge because it was trying to chase down harder, more complex patterns that might not even be real.**

- **Justification for Final Parameter Selections**

C1 : has a better combination of training time and accuracy

Case 2: Multi-Layer Perceptron (MLP) (Network Architecture/Capacity)

- **The Experiment: "Deep/Wide" architecture (300, 150...) vs. a "Shallow/Narrow" one (100, 75...).**
- **The Result: A catastrophic failure in the smaller model.**
    - ***Accuracy:*** **The large model achieved 0.82, while the smaller model collapsed to 0.10.**
    - ***Training Time:*** **The large model took 297s, the small one only 10s.**

Model	Params	Mean Accuracy	Mean F1	Training Time (s)
MLP	**Layers: (300, 150, 100, 50, 10)**	**0.82**	**0.82**	**297.25**
MLP	**Layers: (100, 75, 50, 25, 5)**	**0.10**	**0.02**	**10.47**

WHY??:

- **Model Capacity: This refers to the model's "brain size." The first MLP has layers starting with 300 neurons. The second starts with 100.**
- **The Collapse: An accuracy of (10%) is statistically equivalent to random guessing. This suggests the smaller architecture didn't just perform "worse"—it failed to learn *anything*.**
- **Reasoning: The smaller network likely suffered from underfitting. It simply lacked the mathematical complexity (neurons and connections).**

- **Justification for Final Parameter Selections**

- **This configuration achieved an acceptable accuracy of 82% using a simple feedforward neural network. In comparison, the alternative model produced only 10% accuracy, which is close to random guessing and indicates poor learning capability.**

Requirement 8:

1. ☒**Real-world Applicability**
   - ☒Scalability considerations
   - ☒Potential applications and limitations

Output 8:

A. Scalability Considerations

Scalability in OCR means handling larger volumes of documents (e.g., scanning a 500-page book vs. a single ID card) without crashing or taking too much time.

- **Processing Speed:**
  - **The Data: Logistic Regression model trains very fast (17.77s training), while the Large MLP takes significantly longer (297.25s).**
  - **Real-World Impact: If we are building a mobile document scanner, the MLP might be too heavy to run directly on the phone (Edge AI). However, the Logistic Regression, while fast, only has 74% accuracy. In OCR, speed is useless if the text is unreadable. Then we are forced to use the heavier MLP to get usable results, which means we might need to offload processing to a cloud server to handle scale.**
- **The "Shrink Model" Trap:**
  - **When we tried scale down the MLP to a smaller architecture (100, 75, 50...) to perhaps save resources , Accuracy collapsed to (10%).**
  - **Implication: This proves that for this specific OCR task, we cannot simply "shrink" the model to make it more scalable. There is a minimum complexity floor.**

B. Potential Applications and Limitations

Potential Applications:

1. **Automated Grading Systems:**
   - **Scanning student IDs and scores on bubble sheets or handwritten exam papers.**
   - ***Relevance:* The 0.82 accuracy is useful for a "first pass" aid to teachers, provided human verification is included.**
2. **Utility Meter Reading:**
   - **Reading digits from gas, water, or electricity meters to automate billing.**
3. **Bank Check Processing:**
   - **Recognizing the written amount or account numbers on checks (though this requires extremely high accuracy).**

Limitations (Critical for OCR):

1. **The Accuracy Ceiling (82% is Risky):**
   - **The Data: best model (MLP) only achieved 0.82 (82%) accuracy.**

- **The Limitation: In the world of OCR, 82% is often unacceptable. If you scan a phone number, getting 2 digits wrong renders the output useless. If you scan a contract, an 18% error rate changes the legal meaning. This suggests that while the MLP is better than Logistic Regression, it might still require human review in production.**

2. **Noise Sensitivity:**
    - **The drop in Logistic Regression performance (from 0.74 to 0.69) when changing C suggests the data is sensitive to noise. In the real world, OCR images often have shadows, crinkled paper, or bad lighting. If the model is this sensitive to hyperparameter changes, it may be brittle when facing real-world image noise.**
3. **Confusion of Similar Digits:**

- **The drop in accuracy suggests the model is struggling with specific "confusers":**
    - **1 vs 7: Depending on handwriting (the "cross" on the 7).**
    - **0 vs 6 vs 8: Loop closures.**
    - **5 vs 6: If the top of the 5 is connected.**
- ***Real-world consequence:* Misinterpreting a '1' as a '7' in a bank transaction is a critical failure.**

Requirement 9:

Deliverables

1. ☐**Code Repository**
   - ☐Well-documented implementation of all three models
   - ☐Data preprocessing scripts
   - ☐Evaluation scripts
2. ☐**Technical Report**
   - ☐Experimental results
   - ☐Comparative analysis
   - ☐**Presentation**

Github Link : ML Project

Output 9 :

Model	Params	Mean Accuracy	Mean Precision	Mean Recall	Mean F1	Standard Deviation	Training Time (s)
NaiveBayes	{}	0.51	0.54	0.51	0.51	0.01	0.10
LinearReg_OvA	{}	0.64	0.64	0.64	0.64	0.01	4.24
LogisticReg	{'C': 1.0}	0.74	0.74	0.74	0.74	0.01	17.77
LogisticReg	{'C': 10.0}	0.69	0.69	0.69	0.69	0.01	26.85
MLP	{'hidden_layer_sizes': (300, 150, 100, 50, 10)}	0.82	0.83	0.82	0.82	0.02	297.25
MLP	{'hidden_layer_sizes': (100, 75, 50, 25, 5)}	0.10	0.01	0.10	0.02	0.00	10.47

Conclusion

In this study, we evaluated four distinct modeling approaches to solve the Handwritten Digit Recognition problem. We progressed from a simple probabilistic baseline (Naive) to a Linear Model (OVA), and finally to non-linear Neural Networks (MLP) with varying capacities. This progression allowed us to analyze the trade-off between model complexity, computational cost, and recognition accuracy.

3. Model-by-Model Analysis

1. Naive Model (Baseline)

- **Role: The Naive model served as our probabilistic baseline. It operates on the assumption that all features (pixels) are independent of each other—a strong assumption that does not hold true for image data where neighboring pixels form shapes.**
- **Analysis: As expected, this model provided the lowest performance tier. It acts as a "sanity check" for our experiment; any sophisticated model (Linear or MLP) must significantly outperform this baseline to be considered effective. Its primary advantage is speed, but it lacks the capacity to model the geometric structures of handwritten digits.**

2. Linear OVA (One-Vs-All)

- **Role: We implemented a Linear Model using the One-Vs-All (OVA) strategy to handle the multi-class nature of digits (0-9). This approach trains 10 binary classifiers (one for each digit against the rest).**
- **Analysis: This model improved upon the Naive baseline by assigning weights to pixels. However, it is fundamentally limited by linearity. It attempts to separate digits using straight hyperplanes. Since handwriting varies wildly (e.g., a slanted '1' vs. a straight '1'), a simple linear boundary often fails to capture these variations, leading to a performance ceiling that cannot be broken without non-linear techniques.**

3. Logistic Regression

- **Configuration A (C=1.0): achieved (74%) accuracy.**
- **Configuration B (C=10.0): achieved (69%) accuracy.**
- **Analysis: The results highlight the model's sensitivity to overfitting.**
    - **The model with C=1.0 (stronger regularization) performed better, suggesting that constraining the model helps it generalize to new data.**
    - **When we increased C to 10.0 (reducing regularization), the accuracy dropped to 69%, and training time increased (17s vs 26s). This indicates the model began "memorizing" noise in the training set rather than learning true digit shapes. Despite the tuning, the Logistic approach peaked at 74%, confirming that linear methods are insufficient for high-precision OCR.**

4. Multi-Layer Perceptron Variants

- **Architecture A (Small - 100, 75...): Failed with 0.10 (10%) accuracy.**
- **Architecture B (Large - 300, 150...): Succeeded with 0.82 (82%) accuracy.**
- **Analysis: This comparison provides the most critical insight of the experiment: Capacity Threshold.**
    - **The Small MLP failed completely (10% is random guessing), proving that below a certain complexity, the network cannot learn *any* useful features.**
    - **The Large MLP broke the linear barrier, achieving 82%. The introduction of deep hidden layers allowed the model to learn non-linear hierarchies (edges -> curves -> loops).**
    - **Trade-off: The superior accuracy comes at a cost. The Large MLP took 297.25s to train, compared to just 17.77s for the best Logistic model. However, for a Number Recognition system, the jump from 74% to 82% accuracy is worth the extra computational time.**

Conclusion

The experiment confirms a hierarchy of capability. The Naive and Linear OVA models provide speed but suffer from high bias (underfitting). The Logistic Regression analysis showed that even with hyperparameter tuning, linear models hit a hard ceiling at 74%. The MLP analysis demonstrated that a high-capacity Neural Network is required to solve the non-linear challenges of digit recognition, making the Large MLP (82%) the clear winner for deployment.