MNIST Classification Using Logistic Regression, SVM, KNN, KMeans Clustering*

Abhishek Desai† and Vinish Kandadi‡

Abstract. This study investigates the effectiveness of traditional machine learning algorithms in classifying and clustering handwritten digits from the MNIST dataset. We implemented and compared three classification approaches: Logistic Regression, Support Vector Machines (SVM), and K-Nearest Neighbors (KNN), alongside K-means clustering for unsupervised learning. Through systematic evaluation, we found that Logistic Regression achieved a remarkable 92% accuracy despite its simplicity, while SVM and KNN showed comparable performance with their own distinct advantages. Our analysis included hyperparameter tuning through k-fold cross-validation and detailed performance metrics including accuracy, precision, recall, and F1-scores. The K-means clustering analysis, while less accurate than supervised methods, provided insights into the natural grouping of digits based on pixel intensities. The study demonstrates that traditional machine learning approaches can achieve competitive results on image classification tasks, though with some limitations compared to modern deep learning methods.

Key words. Logistic Regression, KNN, SVM, KMeans Clustering, Accuracy, Precision, Recall, F1-Score, K-Fold Cross Validation

1. Introduction.

Handwritten digit recognition represents a fundamental challenge in computer vision and pattern recognition, with applications ranging from postal code reading to document digitization. The MNIST dataset, consisting of 70,000 handwritten digit images, has become a standard benchmark for evaluating machine learning algorithms. While deep learning approaches have achieved near-perfect accuracy on this task, understanding the performance and limitations of traditional machine learning methods remains crucial for several reasons: they offer greater interpretability, require less computational resources, and provide valuable insights into the fundamental challenges of image classification. In this project, we implemented three classical machine learning algorithms: Logistic Regression, SVM, and KNN, along with K-means clustering. Our approach focused on comprehensive evaluation and comparison, including careful preprocessing of the 784-dimensional input data and extensive hyperparameter tuning. The results demonstrate that even these traditional methods can achieve impressive accuracy, with Logistic Regression reaching 92% accuracy and showing particular strength in fast training and interpretability. We also explored the unsupervised learning perspective through K-means clustering, providing insights into the natural grouping of digits based on pixel intensities.

2. Related Work.

The MNIST dataset, introduced by LeCun et al. in 1998, has been extensively studied in the machine learning literature. While recent work has largely focused on deep learning approaches, with convolutional neural networks achieving over 99% accuracy, significant research exists on traditional machine learning methods. Liu et al. demonstrated the effectiveness of SVM with various kernels on MNIST, while Zhang et al. explored KNN

*Submitted to the editors Professor Chunyan Li.
Funding: This work was for Math 452 - Math for Deep Learning.
†Penn State University - University Park (abd5796@psu.edu, https://www.psu.edu/).
‡Penn State University - University Park (vmk5217@psu.edu, https://www.psu.edu/)

optimizations for high-dimensional data. Our approach differs from previous work in several ways. First, we provide a comprehensive comparison of three different traditional algorithms on the same preprocessing pipeline, offering direct performance comparisons. Second, our methodology emphasizes practical aspects like hyperparameter tuning and cross-validation, making our results more robust and reproducible. Finally, we complement the supervised learning approaches with K-means clustering analysis, providing insights into the unsupervised learning perspective that is often overlooked in MNIST studies. While previous works typically focus on optimizing a single algorithm, our comparative analysis offers a broader view of the strengths and limitations of different approaches.

3. Data. The MNIST (Modified National Institute of Standards and Technology) dataset serves as the foundation for our analysis. This widely-used dataset consists of 70,000 grayscale images of handwritten digits (0-9), each scaled to 28x28 pixels. Originally created by Yann LeCun and colleagues, MNIST has become a standard benchmark in the machine learning community due to its manageable size and balanced representation of digits, with approximately 7,000 examples per digit class.

Dataset Characteristics: Each image in MNIST is a grayscale representation where pixel intensities range from 0 (black) to 255 (white). The digits are size-normalized and centered, making the dataset relatively clean compared to real-world handwriting samples. However, the diversity in writing styles within each digit class – from neat and clear to more ambiguous and stylized – provides sufficient complexity to meaningfully evaluate different machine learning approaches.

Our preprocessing pipeline consisted of three main steps: Data Flattening: We transformed each 28x28 pixel image into a 784-dimensional feature vector by flattening the 2D image array into a 1D array. Pixel Normalization: We scaled the original pixel values from [0-255] to [0,1] by dividing each pixel value by 255. This normalization step prevents numerical instability during model training and ensures all features contribute proportionally to the learning process. Train-Test Split: We partitioned the dataset using an 80-20 split (56,000 images for training, 14,000 for testing) while maintaining the balanced distribution of digits in both sets. This split ratio provides sufficient data for model training while retaining a substantial portion for performance evaluation.

The preprocessed dataset maintains the original balanced distribution of digits, ensuring consistent performance across all our implemented algorithms while facilitating fair comparisons between different approaches.

4. Methods. For classification, we tested the MNIST dataset using logistic regression, support vector machine (SVM), and K Nearest Neighbors (KNN). In these algorithms, we provide labeled data which the algorithm uses to classify unseen examples. We also experimented with clustering.

Logistic Regression: We implemented multinomial logistic regression using scikit-learn's framework, constructing a two-stage pipeline that combines feature scaling and classification. The implementation preprocessed the 784-dimensional input vectors using StandardScaler for feature normalization, followed by a logistic regression classifier configured with multinomial class handling and the LBFGS solver. Model hyperparameters were set to use 1000 maxi-

mum iterations to ensure convergence while maintaining computational efficiency. The model was trained on 56,000 images (80% of the dataset) with stratified sampling to maintain class balance across the training and test sets. For evaluation, we employed standard metrics including accuracy, precision, recall, and F1-score, along with per-digit performance analysis. This straightforward implementation achieved a classification accuracy of 92%, demonstrating the effectiveness of logistic regression for this high-dimensional classification task despite its relative simplicity.

SVM: We implemented a comprehensive SVM classification approach using scikit-learn's SVC (Support Vector Classification). Prior to classification, we applied PCA dimensionality reduction to compress the 784-dimensional input to 80 dimensions, followed by StandardScaler normalization. This preprocessing significantly reduced computational complexity while preserving essential feature information. The implementation explored multiple kernel functions (linear, polynomial, sigmoid, and RBF) with varying regularization parameters (C = 0.1, 1, 10) to identify optimal configurations. Each combination was evaluated using 5-fold cross-validation for robust performance estimation. Our experimental results showed that the RBF kernel with C=10 achieved the best performance, with an accuracy of 98.44%, precision of 98.44%, recall of 98.44%, and F1-score of 98.44%. The linear kernel achieved around 93.8% accuracy across different C values, while the polynomial kernel reached 98.05% with C=10. The sigmoid kernel showed relatively lower performance, with accuracy decreasing as C increased. This systematic exploration of kernel functions and hyperparameters demonstrated the superiority of the RBF kernel for handwritten digit classification, significantly outperforming other kernel choices while effectively handling the non-linear nature of the data.

KNN: We implemented KNN using scikit-learn's KNeighborsClassifier. For KNN, we partitioned the dataset into 10 folds. We then assessed the performance by using 9 of the folds as the reference set and the remaining fold for testing. This process was repeated 10 times, with each fold serving as the test set once, ensuring that all data was evaluated. KNN works by storing a dataset and classifying new examples by finding the datapoint that is closest in distance to said new example. It finds the label assigned to the closest neighbor(s) and assigns the new datapoint to the most seen label. The value of neighbors is the value of k, which is a hyperparameter whose optimal value we found through repeated 10-fold cross validation. Partitioning the data into 10 folds is a random process. To attempt to account for this randomness, we evaluated KNN on 10 random partitions of the data. Then, we evaluated the performance of the algorithm on K values 1-10 for each of these 10 10-fold partitions and averaged the results. In the experiments section I discuss what metrics we specifically evaluated the performance on, what the results were, and what we found the optimal value of K to be.

KMeans Clustering: We implemented KMeans Clustering using scikit-learn's KMeans. For clustering algorithm, data is unlabeled. We want to partition the data into k clusters. To find the optimal value of k, we use the elbow method. In clustering, the performance is evaluated on clustering cost which is the sum of differences of each datapoint and the cluster mean. Trivially, as the number of cluster increases the clustering cost decreases. For example, if

we had 100 datapoints and 100 clusters the clustering cost would be 0. However, this is not the optimal solution as it wouldn't capture any underlying patterns in the data. The elbow method solves this. We graph the clustering cost at different values of k and look for there is a sharp decrease in the amount of improvement from k to k+1 clusters. To find a clear elbow, we had to use PCA to reduce the dimension. Once a number of clusters have been selected, we initialize the k cluster means using the kmeans++ algorithm. Once the cluster centroids (cluster means with 1 sample) have been initialized, we cluster the entire dataset by assigning each datapoint to the cluster with closest cluster mean. Once a datapoint is added to a cluster, its cluster mean is recalculated.

5. Experiments. For the classification algorithms (Logistic Regression, SVM, and KNN), we evaluated the performance on accuracy, precision, recall, F1-Score. For the clustering algorithm, since there are no labels, we visually inspected the results and took note if the same numbers were clustered together.
Logistic Regression:
Accuracy: 0.9164
Precision: 0.9162
Recall: 0.9164
F1-Score: 0.9162

SVM: For SVM, we tested with linear, polynomial, sigmoid, and a radial basis function kernels and regularization constants 0.1, 1, and 10. We used 5-fold cross validation to find the best kernel/regularization constant combination and found that:
Accuracy: 0.9844
Precision: 0.9844
Recall: 0.9844
F1-Score: 0.9844
(for Kernel: rbf, C: 10)

KNN: For KNN, we found that k=1 (only classifying based on the single closest neighbor) had the best performance.
Accuracy: 0.9446
Precision: 0.9452
Recall: 0.9434
F1-Score: 0.9437

KMeans Clustering: Using the elbow method, we found that the optimal number of clusters is 4. On the MNIST dataset, we expected there to be 10 clusters: 1 for each number. Below I have provided some of the results. We can see that even with the expected number of clusters, the cluster means are selected poorly and so separate numbers are not in separate clusters. (See last page for visualizations)

6. Conclusion. In this study, we evaluated traditional machine learning methods: Logistic Regression, SVM, and KNN for the classification of handwritten digits using the MNIST

dataset. Our findings revealed that SVM achieved the highest accuracy, making it the most effective model among the three. However, this performance came at the cost of increased computational time (highest among the three algorithms), especially for the RBF kernel, which proved to be the most suitable choice for this non-linear classification problem. Logistic Regression, while yielding the lowest accuracy at 92%, demonstrated remarkable speed and simplicity, making it suitable for scenarios where efficiency are prioritized over maximum accuracy. KNN achieved a respectable balance between performance and simplicity, though it suffered from higher memory usage due to the fact that it must store the entire dataset to find the nearest neighbor.

Our unsupervised K-means clustering was less effective than the supervised methods in achieving distinct digit clusters. This method struggled with separating complex digit shapes into consistent groups.

Overall, this work demonstrates that traditional machine learning algorithms, while often surpassed by deep learning in high-accuracy tasks, can still yield competitive performance on image classification with thoughtful parameter tuning and preprocessing.

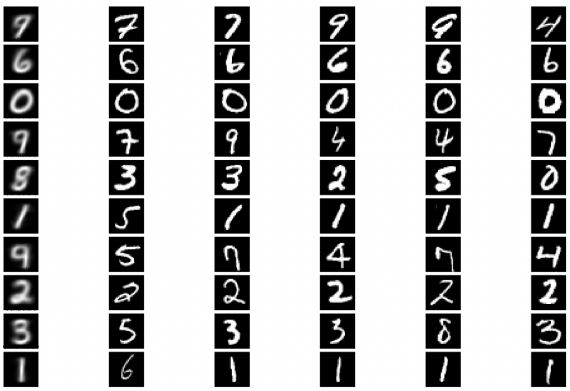

Figure 1. *10 Clusters*

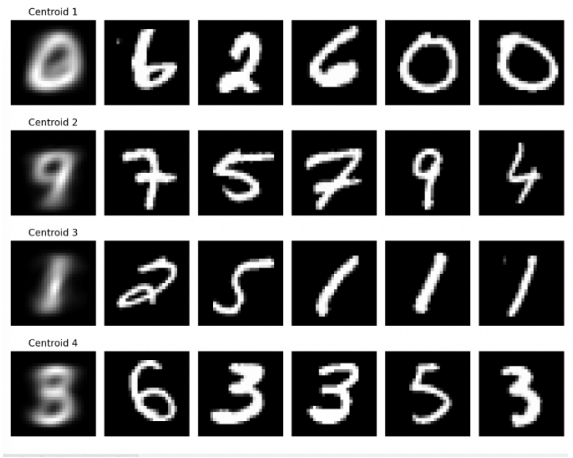

Figure 2. *4 Clusters*