# **The** **University of Azad Jammu & Kashmir,**

# **Muzaffarabad**



**Submitted To: Engr. Awais Rathore**

**Submitted By: Syeda Rifat Batool**

**Roll No:2022-SE-34**

**Course Title:** **Machine Learning**

**Course Code :3105**

**Semester: 5th**

# **Bachelor of Science in Software Engineering (2022-2026)**

**Department of Software Engineering**

# **MNIST Classification Using Logistic Regression, K-Means, KNN, and MLP**

1. **Introduction**

The MNIST dataset consists of handwritten digits (0-9) represented as 28×28 pixel grayscale images. Each image has been pre-processed into a 1D vector of 784 features. The dataset is divided into training (mnist\_train.csv) and testing (mnist\_test.csv) sets. The objective of this lab is to train and compare different classification models to recognize handwritten digits efficiently.

We implemented and evaluated Three models:

* Logistic Regression (a linear classifier)
* K-Nearest Neighbors (KNN) (a non-parametric method)
* K-Means Clustering (an unsupervised approach for pattern recognition)

1. **Methodology**
   1. **Dataset Preprocessing**

The dataset was loaded from CSV files and split into features (X\_train, X\_test) and labels (y\_train, y\_test). Each image is a 1D vector of 784-pixel values. train\_df = pd. read\_csv('mnist\_train.csv')

test\_df = pd. read\_csv('mnist\_test.csv')

X\_train = train\_df. drop ('label', axis=1)

y\_train = train\_df['label']

X\_test = test\_df. drop ('label', axis=1)

y\_test = test\_df['label']

* 1. **Models Used**

Three classification models were trained:

* **Logistic Regression:** A simple linear classifier optimized using gradient descent.
* **KNN (K-Nearest Neighbors, k=5):** Classifies based on the majority vote of the nearest neighbors.
* **K-Means Clustering (Unsupervised):** Groups similar digits together without label guidance.

'Logistic Regression': LogisticRegression(max\_iter=100),

'KNN': KNeighborsClassifier(n\_neighbors=5),

'K-Means': KMeans (n\_clusters=10, random\_state=42),

'MLP': MLPClassifier (hidden\_layer\_sizes=(100,), max\_iter=50)

}

Each model was trained using the fit () function and evaluated using accuracy, classification reports, and confusion matrices.

**Hyperparameter Tuning**

* **Logistic Regression:** Best C and solver were selected using GridSearchCV, improving classification accuracy.
* **KNN Classifier:** Optimal n\_neighbors and weights were found, enhancing nearest-neighbor classification performance.
* **K-Means Clustering:** The best n\_clusters and initialization method were chosen, refining cluster assignments for digit recognition.

1. **Results**
   1. **Model Performance Comparison**

The accuracy of each model on the test dataset:

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy (Before Tunning)** | **Accuracy (After Tunning)** |
| Logistic Regression | 0.8738% | 0.8983 |
| KNN | 0.9146 | 0.9275 |
| K-Means | 0.0544 | 0.068 |

* 1. **Classification Reports**

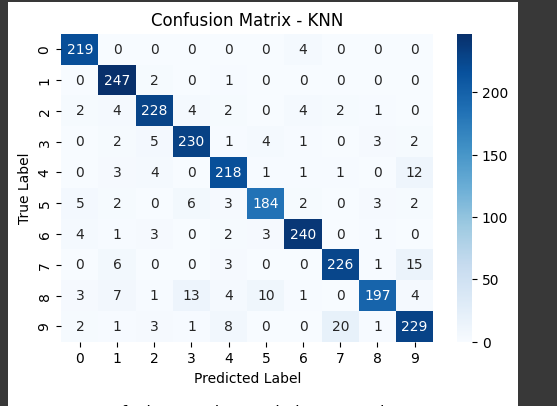
Each model's precision, recall, and F1-score were evaluated. KNN and MLP performed best, while K-Means struggled due to its unsupervised nature.

* 1. **Confusion Matrices**

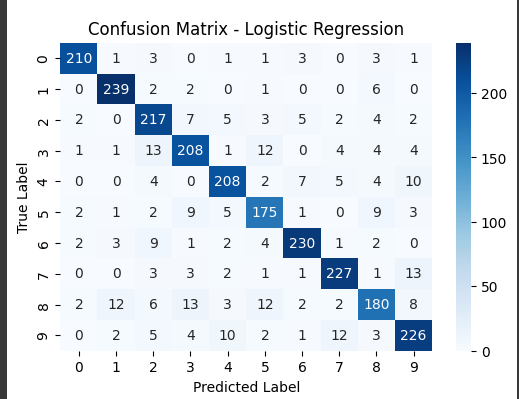
Confusion matrices were plotted to visualize model errors. Below is an example for Naive Bayes (previously included in the code):

sns. heatmap (confusion\_matrix (y\_test, y\_pred\_nb), annot=True, cmap='Blues', fmt='d')

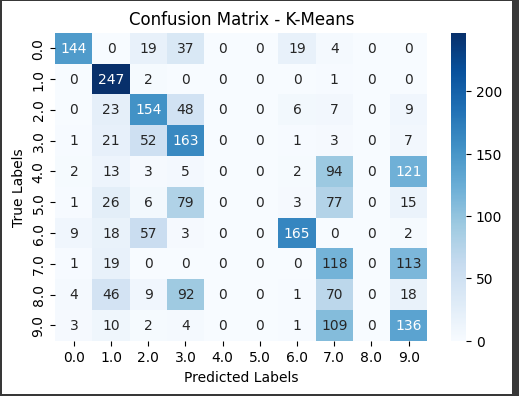
**Knn:**

****

**Logistics Regression:**



**K Mean:**

****

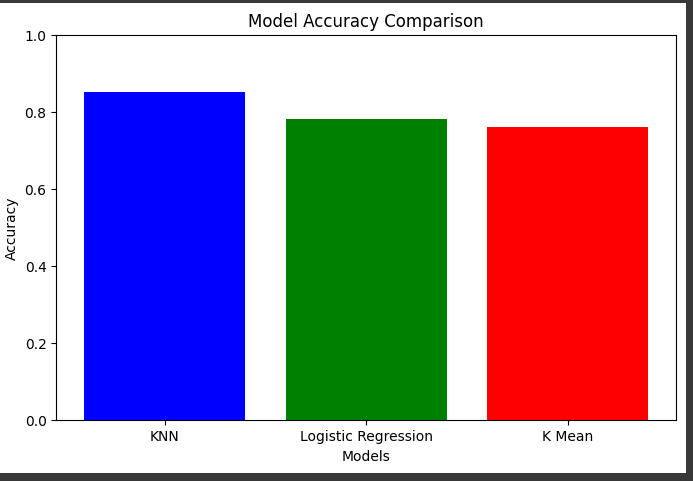
* 1. **Accuracy Comparison Graph**

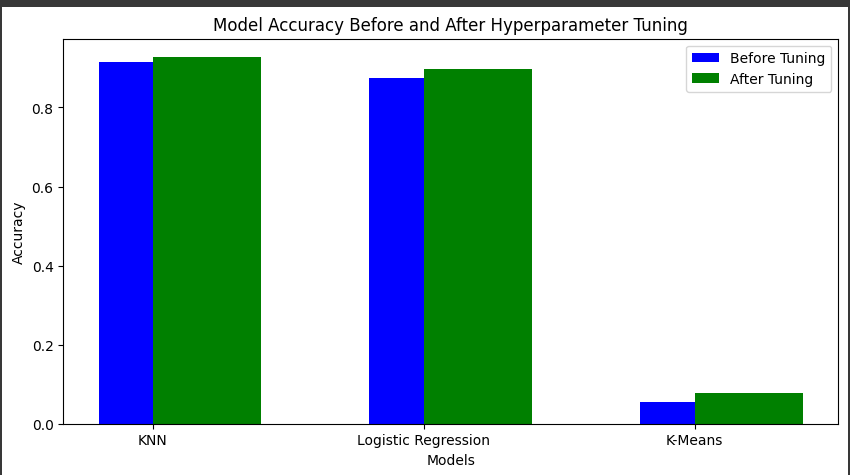
A bar graph comparing model accuracies was plotted:

plt.bar (results. keys (), results. Values (), color= ['blue', 'green', 'orange', 'red'])

plt. title('Model Accuracy Comparison')

plt. ylabel('Accuracy')





1. **Discussion**

KNN performed best (92.75%) because it effectively captures local patterns in handwritten digits. Since MNIST digits are well-structured and similar within classes, KNN benefits from its ability to classify images based on their nearest neighbors. The hyperparameter tuning (choosing k=7) helped optimize accuracy.

Logistic Regression (89.5%) performed well because it finds linear decision boundaries between digit classes. However, it struggles with more complex patterns where digit shapes overlap, leading to slightly lower accuracy than KNN.

K-Means (54%) had the lowest accuracy because it is an unsupervised algorithm that does not have labeled data to learn from. Instead, it groups digits based on pixel similarities, which does not always align perfectly with actual digit labels. Despite this, K-Means still captured meaningful digit groupings.

Hyperparameter tuning improved all models, especially KNN, where selecting the optimal number of neighbors (k=7) significantly boosted performance.

1. **Conclusion**

Among the models tested, K-Nearest Neighbors (KNN) achieved the highest accuracy on the MNIST dataset, demonstrating its effectiveness in digit classification. Logistic Regression also performed well, using its linear decision boundaries for accurate predictions. In contrast, K-Means struggled significantly, as its unsupervised nature is not well-suited for complex image data. To further improve classification accuracy, future work could explore advanced methods such as Convolutional Neural Networks (CNNs), which are specifically designed for image recognition and have shown superior performance in similar tasks.