# Handwritten Digit Recognition with Support Vector Machines (SVM)

**PROJECT REPORT**

## 1. Introduction:

Handwritten digit recognition presents a well-known challenge in the domains of machine learning and computer vision. The objective of this project is to create a system capable of accurately categorising handwritten digits (0-9) from images. Our approach involves employing Support Vector Machine (SVM), a robust supervised learning technique renowned for its proficiency in classification tasks. SVM operates by identifying the ideal hyperplane that effectively segregates data into distinct classes while maximising the margin between these classes.

## 2. Dataset:

The MNIST dataset is a highly utilised benchmark dataset within the realms of machine learning and computer vision. It comprises an extensive collection of grayscale images measuring 28x28 pixels, each depicting handwritten digits ranging from 0 to 9, along with corresponding labels indicating the represented digit.

This dataset is distributed across two files:

- **mnist_train.csv:** This file encompasses 60,000 examples and their respective labels, designed for training purposes.

- **mnist_test.csv:** Consisting of 10,000 test examples and their labels, this file serves the purpose of evaluating model performance.

Each row within these files is structured as follows: the first value represents the label, denoting a number from 0 to 9, while the subsequent 784 values represent pixel intensities, ranging from 0 to 255. Each pixel value characterises the darkness or lightness of the corresponding pixel in the image.

## 3. Data Preprocessing:

The preprocessing of the data involved several steps to ensure its quality and prepare it for model training. Here's a breakdown of the process:

- **Descriptive Statistics:**

Descriptive statistics were calculated to gain insights into the characteristics of the features in the dataset. This included measures such as mean, median, standard deviation, minimum, and maximum values for numerical features. These statistics provided an overview of the central tendency, spread, and distribution of the data.

- **Checking for Null Values:**
  Null values, if present, were identified and handled appropriately. This may involve imputation techniques such as replacing null values with the mean, median, or mode of the respective feature, or removing rows or columns with null values, depending on the significance of the missing data.

- **Analysing Distribution of Samples per Label:**
  The distribution of samples across different class labels was examined to ensure that the dataset is balanced or to identify any class imbalance issues. Class imbalance can impact the performance of machine learning models, so techniques such as oversampling, undersampling, or class weight adjustments may be employed to address this issue.

- **Visualizing Pixel Value Frequencies:**
  Pixel value frequencies were visualized to understand the distribution of pixel values across the dataset. This visualization helps in identifying any patterns or anomalies in the pixel values that may require further investigation or preprocessing.

- **Performing Sample Visualizations:**
  Sample visualizations were generated to get a visual understanding of the data. This may include displaying a random subset of images or data points from the dataset along with their corresponding labels. Sample visualizations help in verifying the correctness of the data and gaining insights into the nature of the problem.

- **Scaling the Dataset:**
  Finally, the dataset was scaled to ensure uniformity across features. Feature scaling is important, especially for algorithms sensitive to feature magnitudes, such as SVMs. Common scaling techniques include standardization (subtracting the mean and dividing by the standard deviation) or min-max scaling (scaling features to a specified range, usually between 0 and 1).

By following these preprocessing steps, the dataset is cleaned, structured, and prepared for model training, ensuring that the machine learning models can learn effectively from the data.

## 4. Model Training:

Three SVM models were trained and compared:
Certainly, let's expand on each model:

**i) Linear SVM:**
  Linear SVM is a variant of the Support Vector Machine (SVM) algorithm designed to separate classes using a straight line in the feature space. It operates by computing the dot product between feature vectors, resulting in a linear decision boundary. This makes it ideal for datasets where classes are easily separable by a straight line or plane. Linear SVM is known for its computational efficiency and ease of interpretation, making it suitable for tasks with high-dimensional feature spaces or large datasets. Its decision boundary is straightforward to visualize and comprehend, making it valuable for understanding the underlying data patterns.

**ii) Polynomial SVM:**

Polynomial SVM extends the capabilities of SVM by utilizing polynomial kernel functions to map the input space into a higher-dimensional feature space. Unlike Linear SVM, Polynomial SVM can capture non-linear relationships between features and class labels by introducing polynomial terms. This enables it to model complex patterns in the data, making it suitable for datasets with non-linear decision boundaries. However, the complexity of Polynomial SVM increases with the degree of the polynomial kernel, potentially leading to overfitting and higher computational costs. Despite these challenges, Polynomial SVM remains a valuable tool for classification tasks that require the discrimination of non-linearly separable classes.

**iii) RBF SVM (Radial Basis Function SVM):**

RBF SVM is another variant of the SVM algorithm that employs a radial basis function (RBF) kernel to map the input space into an infinite-dimensional feature space. This allows RBF SVM to capture complex and non-linear relationships between features and class labels, resulting in highly flexible decision boundaries. The Gaussian nature of the RBF kernel enables RBF SVM to adapt to intricate patterns in the data, making it suitable for a wide range of classification tasks, including those with overlapping classes and non-linear decision boundaries. However, the interpretability of RBF SVM may be limited due to the complexity of its decision boundary, especially in high-dimensional feature spaces or with large datasets. Nonetheless, RBF SVM remains a powerful and versatile algorithm for classification problems requiring robust generalization capabilities.

Performance evaluation metrics such as accuracy, confusion matrix, precision, recall, and F1-score were computed for each model.

## 5. Hyperparameter Tuning:

Hyperparameter tuning is a crucial step in optimizing machine learning models for better performance. In the case of the RBF SVM model, grid search technique was employed to systematically explore a range of hyperparameters. Through K-fold cross-validation, where the dataset is divided into K subsets and the model is trained and validated K times, optimal hyperparameters were identified as “C=10” and “Gamma=0.001”. C represents the regularization parameter, controlling the trade-off between maximizing the margin and minimizing the classification error, while Gamma defines the influence of a single training example, with lower values implying a broader influence. By selecting these hyperparameters, the RBF SVM model is fine-tuned to strike the optimal balance between model complexity and generalization ability, enhancing its predictive performance on unseen data.

## 6. Final Model Building and Evaluation:

A final RBF SVM model was constructed using the best hyperparameters and evaluated on the test dataset. It achieved an accuracy of approximately 97.33%. Confusion matrix and class-wise metrics provided insights into the model's performance.

## 7. Results:

The accuracy of the three models are as below:
- Linear SVM: Accuracy - 92.93%
- Polynomial SVM: Accuracy - 96.11%
- RBF SVM: Accuracy - 96.6%

The final RBF SVM model achieved an accuracy of 97.33% on the test dataset.

## 8. Conclusion:

In conclusion, this project underscores the efficacy of Support Vector Machines (SVM) in the realm of handwritten digit recognition tasks. Notably, the model showcased commendable accuracy and exhibited robust performance throughout its evaluation. While the current study establishes SVM as a viable solution, future endeavors may seek to explore advanced methodologies, such as the utilization of Convolutional Neural Networks (CNNs), to further enhance accuracy and efficacy. By integrating these state-of-the-art techniques, the project not only bolsters advancements in image processing and pattern recognition domains but also offers a dependable framework for addressing handwritten digit recognition challenges. This work thus stands as a testament to the continued evolution and refinement of machine learning methodologies in addressing real-world classification tasks.