Intro to Machine Learning
Project 3

MNIST is a classification dataset where the target variable is categorical (0 to 9). Hence classification algorithms are used to train this dataset.

The following approaches are used to handle the given data.

1. Logistic Regression: This approach is chosen for its simplicity and ease of interpretation, making it an ideal starting point for classification tasks. While it may not capture complex patterns like more advanced models, it provides a basic understanding of the data.
2. Support Vector Machines (SVMs): This algorithm is suitable for the non-linear and complex datasets. Their ability to capture intricate patterns promises high classification accuracy.
3. Random Forests: It is an ensemble learning method that can handle complexity in the data. It can avoid focusing too much on small details and give more reliable predictions preventing overfitting.
4. Convolutional Neural Networks (CNNs): These are ideal for image datasets because they're designed to capture complex relationships within images. With specialized layers like convolutional and pooling layers, CNNs excel at capturing spatial patterns, resulting in exceptional accuracy on datasets like MNIST.

Data Processing:

1. Null values were checked and found absent in the train and test dataset.
2. Normalization was applied to the input data to ensure uniform pixel value scales, promoting stable training and enhancing the model's ability to generalize. Initially, a simple neural network model attained a score of 0.8866, which increased to 0.9259 post-normalization. This demonstrates the efficacy of normalization in enhancing accuracy.
3. Flattening converted 2D image matrices (28x28 pixels) into 1D arrays (784 pixels), essential for processing by both traditional and complex machine learning models.

Basic Model Training:

The models were trained with the default parameters and the accuracy scores are presented below:

1. Logistic Regression: 0.9258.
2. Support Vector Machine (SVM): 0.9792
3. Random Forest: 0.9704
4. CNN: 0.9914

Parameter Tuning:

1. Logistic Regression:

- Cross validation: Randomized Search CV was performed on this model by tuning ‘C’ and ‘penalty’. ‘C’ is a parameter that controls the strength of regularization, with smaller values indicating stronger regularization. The list of 'C' values spans from lower to upper bounds, facilitating the algorithm's search for the optimal trade-off between bias and variance by testing models with varying complexities. The penalty parameter specifies the type of regularization, where 'l1' corresponds to Lasso regularization and 'l2' corresponds to Ridge regularization. RandomizedSearchCV randomly tests a limited number of hyperparameter combinations. With n_iter=10 and cv=3, it tries out 10 combinations using 3-fold cross-validation. The best logistic regression model is then stored in the best_estimator_ attribute of the RandomizedSearchCV object. Finally, the accuracy score, classification report, and confusion matrix are printed to evaluate the performance of the model on the test data.

- Results and Analysis:
  The model achieved an accuracy of approximately 92.53% on the test data, correctly predicting digit labels for most samples. Precision ranges from approximately 88% to 96%, implying that predictions are accurate around 93% of the time for specific digits. Similarly, recall varies from approximately 87% to 98%, meaning the model identifies true labels about 98% of the time for certain digits. F1-score, reflecting the balance between precision and recall, ranges from approximately 88% to 97%, indicating the model's overall effectiveness.

  The confusion matrix highlights the model's ability to accurately classify digits, with high counts along the diagonal indicating correct predictions. Although misclassifications occur, they are sparse, primarily between visually similar digits. Overall, the confusion matrix underscores the model's effectiveness in distinguishing between digits, with minimal confusion between select pairs.

  The difference between the default and cross-validated accuracy scores is negligible (0.0005). It suggests that the default parameters of logistic regression already perform reasonably well on this dataset, and further hyperparameter tuning did not significantly improve the model's performance.

  ★ Best Hyperparameters:
  The best hyperparameters {'penalty': 'l2', 'C': 0.1} indicate optimal performance with L2 regularization and a regularization strength of 0.1. The NaN mean test scores for 'l1' regularization in the cross-validation results are due to the 'lbfgs' solver's

incompatibility, as it supports only 'l2' or 'None' penalties. Since no solver was specified during training, the default 'lbfgs' solver was used.

	param_C	param_penalty	mean_test_score
0	0.1	l1	nan
1	100	l2	0.9160166666666667
2	0.001	l1	nan
3	0.001	l2	0.8866999999999999
4	10	l1	nan
5	1	l2	0.9181333333333334
6	10	l2	0.9166
7	1	l1	nan
8	0.01	l1	nan
9	0.01	l2	0.9108

2. Support Vector Machine (SVM):

- Cross Validation:
  A Support Vector Machine (SVM) classifier was trained using the SVM implementation from scikit-learn. Hyperparameter tuning was performed using RandomizedSearchCV, focusing on tuning the 'C' parameter and the choice of kernel function. The 'C' parameter is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error. It helps prevent overfitting by penalizing misclassifications. The kernel parameter determines the type of kernel function used for classification, including 'linear', 'rbf' (Gaussian Radial Basis Function), and 'sigmoid' kernels. RandomizedSearchCV was configured to try out 6 different combinations of hyperparameters, randomly sampling from the specified parameter distributions.

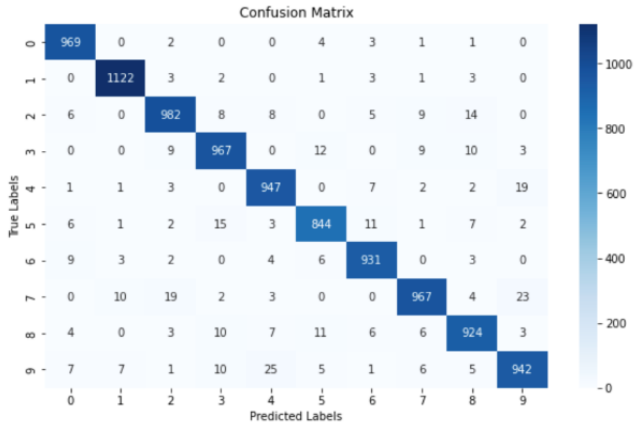
- Results and Analysis:
  The trained SVM model achieved an accuracy of approximately 95.95% on the test data, indicating strong performance in classifying digit labels. The precision, recall, and F1-score metrics are consistently high across different digits, ranging from approximately 94% to 98%. This suggests that the model's predictions are accurate and it can effectively identify true positive instances for various digits.

  The confusion matrix further illustrates the model's performance, showing high counts along the diagonal, indicating correct predictions. Misclassifications are present, but they are relatively sparse and primarily occur between visually similar digits. Overall, the confusion matrix demonstrates the model's ability to differentiate between digits with minimal confusion.

  The default SVM model achieved a higher accuracy (0.9792) compared to the cross-validated model (0.9595). It's possible that the default parameters already captured the underlying patterns in the data effectively. Even though the cross-validated accuracy slightly decreased compared to the default accuracy, it's essential to

consider factors such as overfitting and the potential for improved generalization when selecting the best parameters. Therefore, the parameters identified through hyperparameter tuning are preferred over default parameters to ensure the model's robustness and effectiveness across different datasets.

★ Best Hyperparameters:
The best hyperparameters selected through RandomizedSearchCV are {'kernel': 'rbf', 'C': 0.1}. This indicates that the SVM model achieves optimal performance with an RBF kernel and a regularization parameter 'C' of 0.1. The selection of the RBF kernel suggests that the data might exhibit nonlinear patterns, which the RBF kernel effectively captures, making it the preferred option for achieving superior classification performance.



	param_C	param_kernel	mean_test_score
0	1	sigmoid	0.77775
1	10	linear	0.9202833333333333
2	1	linear	0.9319166666666666
3	0.1	linear	0.93995
4	0.1	rbf	0.9517666666666665
5	0.1	sigmoid	0.8505833333333334

3. Random Forest Classifier:

- Model Training and Tuning:
For the Random Forest classifier, hyperparameter tuning was conducted using Randomized Search Cross-Validation (RandomizedSearchCV). The key hyperparameters tuned were 'n_estimators', 'max_depth', 'min_samples_split', and 'min_samples_leaf'.

a) <u>n_estimators:</u>
The choice of 100, 200, and 300 for the number of trees in the forest allows us to explore a range of model complexities. With fewer trees (100), the model might be simpler but could suffer from high variance. On the other hand, with more trees (300), the model might be more complex and potentially overfit the training data. By testing these values, we aim to find an optimal balance between model complexity and computational efficiency.
b) <u>max_depth:</u>
By testing depths of None, 10, 20, and 30, we investigate the effect of tree depth on the model's performance. A shallow tree (e.g., depth=10) might capture only the

most obvious patterns in the data, while a deeper tree (e.g., depth=30) can capture more intricate relationships. However, allowing trees to grow too deep can lead to overfitting. Therefore, including 'None' allows the trees to grow until all leaves are pure or contain less than the minimum samples required for a split, helping to prevent overfitting.

c) min_samples_split and min_samples_leaf:

These parameters control the minimum number of samples required to split an internal node and to be at a leaf node, respectively. Setting them to 2, 5, and 10 for min_samples_split, and 1, 2, and 4 for min_samples_leaf, allows us to explore a range of granularity in the tree structure. Smaller values can lead to more complex trees that capture finer patterns in the data. However, setting them too low can increase the risk of overfitting by allowing the model to memorize noise in the training data.

Cross-Validation:

RandomizedSearchCV was configured with n_iter=10 and cv=3. This means that 10 combinations of hyperparameters were randomly sampled and evaluated using 3-fold cross-validation.

- **Results and Analysis**:

  Among the 10 combinations tested, the model achieved an accuracy of 97.1% on the test set. While 'n_estimators' didn't significantly impact performance, deeper trees ('max_depth' of 20 or 30) generally outperformed shallower ones. Lower values for 'min_samples_split' and 'min_samples_leaf' were favorable, with the best mean test score of approximately 96.85% obtained with 'n_estimators' of 200, 'max_depth' of 30, 'min_samples_split' of 2, and 'min_samples_leaf' of 1. This suggests that a forest with 200 trees, each with a maximum depth of 30 and minimum samples required to split an internal node and be at a leaf node set to 2 and 1 respectively, performs optimally.

  ★ Best Hyperparameters:

  The best parameters for the Random Forest classifier, derived from hyperparameter tuning, are {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_depth': 30}, optimizing model performance on the given dataset.

  It was observed that most of the best parameters are similar to default parameters hence accuracy did not change even after cross validation.

	param_n_estimators	param_max_depth	param_min_samples_split	param_min_samples_leaf	mean_test_score
0	300		5	1	0.9657
1	100	30	2	2	0.9625666666666666
2	200	20	10	2	0.9614333333333334
3	100	20	10	1	0.9613999999999999
4	100		2	1	0.96525
5	300	10	5	1	0.9448500000000001
6	200		5	1	0.9652333333333334
7	100	20	2	4	0.9592833333333335
8	100	20	5	4	0.9599333333333333
9	200	20	5	2	0.9635666666666666

4. Convolutional Neural Network:

- Model Architecture: The CNN model architecture consists of three convolutional layers for feature extraction, followed by max-pooling layers to downsample feature maps and mitigate overfitting. Two dense layers at the end perform classification based on the learned features, combining high-level representations from convolutional layers to predict input image classes. This design is ideal for digit recognition on the MNIST dataset, enabling the model to learn hierarchical features and make accurate predictions.

- Parameters: The Adam optimizer facilitates efficient weight updates by dynamically adjusting learning rates based on gradient magnitudes, aiding swift convergence and model performance enhancement. Sparse_categorical_crossentropy loss function, designed for multi-class classification tasks, penalizes deviations from true integer labels, promoting accurate predictions. Activation functions (tanh, sigmoid, ReLU) and varied neuron counts in the dense layer (16, 32, 64) optimize the model's ability to capture intricate patterns while mitigating overfitting through regularization.

- The training process involved iterating over the training data for 5 epochs with a batch size of 64. During training, both the training and validation accuracy scores were monitored to assess the model's performance.

- The CNN model showed varying performance across activation functions and neuron counts. 'Tanh' achieved the highest accuracy (98.76% to 99.02%) with 64 neurons, followed by 'relu' (98.86% to 99.02%) with 16 neurons. 'Sigmoid' had slightly lower accuracies (98.22% to 98.47%). Overall, 'tanh' and 'relu' outperformed 'sigmoid'. The combination of 'tanh' activation with 64 neurons yielded the best accuracy (98.76% to 99.02%), making it the top-performing configuration.

Conclusion: Based on the experimental trials on the MNIST dataset, Convolutional Neural Networks (CNNs) emerged as the top-performing model, achieving the highest accuracy, while Logistic Regression, Support Vector Machines, and Random Forests also demonstrated competitive performance, providing alternative options depending on specific requirements and constraints.