CST3170 Final Coursework

Alam Abraham Rincon Rodrigues
Student ID #M00774667

Contents

Description and Aim

This project is focused on implementing and evaluating machine learning models to classify handwritten digits using the Optical Recognition of Handwritten Digits dataset, sourced from the UCI Machine Learning Repository. The study in this project has an additional aim to compare the accuracy of data classification achieved by three different machine learning approaches: Support Vector Machines (SVM), Feedforward Neural Network (FNN), and K-Nearest Neighbour (KNN). Other choices of a machine learning approach could vary according to the developer's preference or interest. These choices will have different accuracy results when implemented. Therefore, the study of several machine learning systems could potentially help decide the most appropriate approach for the presented problem. The incorporation of hyperparameters, k-fold cross-validation, confusion matrices and performance metrics in machine learning systems will also be essential to monitor and evaluate the models' performance.

Machine Learning Application

Programme Manager

In majority of Java projects, it is a good practice and on many occasions, a crucial role for the overall functionality of the application. Behaving as the central coordination of the project, managing the flow of execution of each component in the system, ensuring that all the components (data loading and normalisation, one-hot encoding labels, algorithm selection, hyperparameters tuning, model training and evaluation, and best result's analysis for the algorithm selected) are executed automatically.

Loading The Data

Data manipulation commonly occurs before loading the data. In this project, the datasets are prepared by transforming the data into a suitable format for machine learning.

Normalisation

The data is normalised by scaling pixel intensity values to the range [0, 1], which standardises the data and ensures all features contribute proportionally during the learning process. Normalisation helps improve model convergence by preventing features with larger ranges from dominating the training process and ensures compatibility with requirements of gradient-based optimisation algorithms.

One-Hot Encoding

The label values are also extracted from the datasets and transformed into one-hot encoded vectors, a technique that represents each label as a binary vector, where the index corresponding to the correct class is set to 1, and all other indices are set to 0. One-hot encoding is important for multi-class classification tasks, because it allows the model to compute categorical loss functions and interpret outputs as probabilities for each class.

Hyperparameters

Hyperparameters are predefined settings that control the behaviour and performance of machine learning algorithms. They differ from model parameters, which are learned from the data during training. The hyperparameter combinations in this project are generated and managed systematically for each machine learning system (SVN, FNN and KNN).

Gamma (SVM)

It's a Kernel parameter that defines how far the influence of a single training example reaches. Using low Gamma values, the model considers a wider range of data points to influence the decision boundary while high Gamma values make the SVM focus only on nearby data points.

C (SVM)

It's a regularisation parameter that determines how much importance is given to correct classifying training data versus creating a simple decision boundary. A smaller C value permits the SVM to allow some misclassification in the training data to achieve a smoother decision boundary, focusing on the overall trend of the data rather than memorising specific data points. Larger C values makes the SVM prioritise correct classification of the training data, even if it means creating more complex decision boundary that closely fits the data.

Maximum Iterations (SVM)

It's a parameter that specifies the upper limit on the number of iterations the optimisation algorithm can execute during the training phase, ensuring that the training process terminates within a reasonable timeframe to help balancing the computational efficiency and SVM performance. If the limit is too low the algorithm could stop before finding an optimal solution, and if the limit is too high it would implement unnecessary computations without significant gains in accuracy.

Tolerance (SVM)

It's a parameter that controls when the stopping criterion for the optimisation process. A smaller tolerance value ensures more precise solutions but sacrifices computational time. Larger tolerance value can speed up training, but it could result in less accuracy.

Hidden Neurons (FNN)

It determines the number of units in the hidden layer and influences the FNN's capacity to learn complex patterns. A larger number of hidden neurons increases the ability to capture intricate relationships between the data. While a small amount, will result in the FNN not representing the data adequately.

Epochs (FNN)

It's the number of times the entire dataset is passed through the FNN during training. More epochs allow the FNN to refine the weights through iterative updates, potentially improving accuracy. Although too many epochs may lead to the FNN to become overly tailored to the training data.

Learning Rates (FNN)

It controls the step size of weight updates during training. A high learning rate can speed up convergence but risks overshooting the optimal solution. A low learning rate allows more precise updates, but it could result in slower training.

K (KNN)

It is the number of nearest neighbours considered when classifying a data point. Smaller K values make the KNN sensitive to noise, because the classification depends on fewer data points. Larger K values reduce sensitivity to noise and provide more generalised classifications, but they may blur distinct boundaries between classes.

K-Fold Cross-Validation

K-Fold Cross-Validation is an important method in machine learning for testing how well a model can make predictions on new and unseen data. The K value in the Cross-Validation process stand for the number of equal parts the dataset will be split into. It is a useful process because it reduces the chances of the model working well on one specific split of the data and then perform poorly on others. The result of all the folds are averaged providing a final accuracy to determine how well the model did.

Confusion Matrices and Performance Metrics

Confusion matrices provide a detailed breakdown of the model predictions by counting the true positives, true negatives, false positives and false negatives, for each class, allowing a deep understanding of the model's strengths and weaknesses, mainly in classification task for multiple classes. The performance metrics derived from the confusion matrix are precision, recall, and F1-score. These help by assessing the model performance beyond simple accuracy, especially in cases where accuracy alone cannot provide a complete understanding of imbalanced datasets.

Precision

Is the proportion of correctly predicted positive observations to the total predicted positive observations, meaning that high precision indicates that the model makes few false positive errors. The formula for Precision is $Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$

Recall

Is the proportion of correctly predicted positive observations to all actual positive observations, meaning that high recall means the model captures most of the actual positives

but might include some false positives. The formula for Recall is $Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$

F1-Score

Is the model's balance between precision and recall, useful in imbalance datasets. Meaning that high F1-score indicates that the model has both, good precision and good recall. The formula for F1-Score is $F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

Machine Learning Systems

Machine Learning Systems offer the capabilities to analyse, predict and classify data with high accuracy. To solve complex classification problems, machine learning system can learn patterns from the data by using different algorithms, in this project, the algorithms are Support Vector Machines (SVM), Feedforward Neural Networks (FNN), and K-Nearest Neighbours(KNN).

Support Vector Machines (SVM)

SVM are a flexible classification algorithm, particularly effective in high-dimensional spaces. In this project, the SVM is implemented with an RBF(Radial Basic Function) kernel, and using a one-vs-one classification strategy. The SVM handles the classification problems by decomposing the task into multiple binary classifications.

One Vs One

This approach trains a separate binary SVM classifier for each pair of labels in the dataset. For a dataset with N labels, this results in $\frac{N(N-1)}{2}$ classifiers. During testing, the SVM uses a voting mechanism, where each classifier predicts one of its two labels, and the label receiving the most votes is chosen as the prediction.

Filtering The Data and Labels

Filtering the data was important to include only the instances corresponding to the two labels being considered by the current binary classifier, simplifying the optimisation process by reducing the dataset size.

Predictions and Matrix Updates

During the testing, the predictions from all the binary classifiers are combined using the voting mechanism. For each instance, the true label and predicted label are used to update the confusion matrix.

Binary SVM

The binary SVM is the foundation of the one-vs-one strategy, solving individual binary classification problems. The decision boundary is defined as: $f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b$

Where α_i are the Lagrange multipliers, y_i represents the class labels (+1 or -1), $K(x, x_i)$ is the kernel function and b is the bias term. The optimisation process adjusts α_i to maximise the margin between the two classes.

Radial Basis Function (RBF) Kernel

The RBF kernel handles non-linear class boundaries by projecting data into a higher-dimensional space. It calculates the similarity between two data points based on their

Euclidean distance, allowing the SVM to capture intricate patterns in the data. The functions is $K(x, x') = \exp(-Gamma\|x - x'\|^2)$

Lagrange Multipliers

Used to optimise the decision boundary. Each α_i determines the weight of its corresponding data point in defining the hyperplane. Data points with non-zero α_i values, known as support vectors, shape the decision boundary.

Sequential Minimal Optimisation (SMO)

The SMO algorithm solves the dual optimisation problem, by iteratively updating pairs of Lagrange multipliers (α) while ensuring that all constrains are satisfied.

Karush-Kuhn-Tucker (KKT) Conditions

KKT conditions are used to verify the optimality of the solution in SVM training, ensuring that data points far from the margin have $\alpha_i = 0$, support vectors on the margin have $0 < \alpha_i < C$ where C is the regularisation parameter, and that misclassified points have $\alpha_i = C$ to help maintain the balance between maximising the margin and penalising misclassifications.

Feedforward Neural Network (FNN)

FNN are a type of artificial neural network, also known as Multilayer Perceptrons (MLP). In this project, the FNN is implemented with a single hidden layer and uses the ReLU activation function in the hidden layer for non-linearity. The output layer applies the Softmax function to generate probabilities for multi-class classification. The FNN learns to associate inputs with outputs by adjusting weights through forward and backward propagation. The FNN optimises its performance by iteratively minimising the error between predicted and actual outputs using a combination of weight updates and hyperparameters such as learning rate, hidden layer size, and epochs.

Weights

In this project, weights are initialised with small random values to break symmetry, allowing diverse updates during training. The network uses two sets of weight matrices, one between the input and hidden layers, and another one between the hidden and output layers. These weights are updated iteratively during training to minimise prediction errors.

Forward Propagation

Forward propagation computes activation through the network's layers. Input features are multiplied by weights and passed through the ReLU activation function in the hidden layer for non-linearity and the vanishing gradient problem. The output layer applies the Softmax function, converting raw scores into probabilities.

Vanishing Gradient Problem

Occurs during backward propagation in deep neural networks, where the gradients of weighs in earlier layers become extremely small when activation functions like sigmoid or tanh are used because their derivatives shrink as inputs move away from zero. Consequently, weight updates in these layers become negligible slowing or halting training. ReLU activation function solves this issue by having a derivative of 1 for positive inputs, allowing gradients to flow effectively through the network.

Backward Propagation

Backward Propagation adjusts the weights by propagating the prediction error back through the network. Calculates the output error using the difference between the predicted probabilities and the actual labels. This error is propagated to the hidden layer, where weight updates are computed using the derivative of the ReLU activation function. The updated weights minimise the cross-entropy loss function.

Cross-Entropy Loss

Measures the difference between the predicted probability distribution and the actual labels. It is effective for classification tasks because it penalises confident but incorrect predictions more heavily. Minimising cross-entropy loss, the FNN learns to align its predicted probabilities closely with the true class labels, leading to more accurate predictions.

Rectified Linear Unit (ReLU) and Derivative

The ReLU activation function is used in the hidden layer to ensure computational efficiency and better gradient flow. It's derivative, simplifies weight updates during backward propagation.

The ReLU formula is: $ReLU(x) = \max(0, x)$

The derivative ReLU formula is: $ReLU'(x) = \begin{cases} 1 \; if \; x > 0 \\ 0 \; otherwise \end{cases}$

Softmax

The Softmax function is applied at the output layer to normalise raw activations into probabilities, ensuring that the predictions are interpretable and suitable for multi-class classification problems, where each class is assigned a probability.

K Nearest Neighbour (KNN)

KNN is a simple but effective classification algorithm, known for its instance-based learning approach. In this project, KNN is implemented using Euclidean distance metric to measure similarity between data points and identify the nearest neighbours. The algorithm predicts the label of a given data point by considering the majority label among its k-nearest neighbour.

Most Common Label

This method helps to accurately identify the label that appears most frequently among the neighbours, ensuring the robust classification. The Most Common Label formula is:
$Most\ Common\ Label = \arg\max_{label}(Count(label))$

Where $Count(label)$ is the number of times a label appears among the k-nearest neighbours.

Bubble Sort

The bubble sort method helps identify the nearest neighbours, by calculating and sorting the distances between a test data point and all training points. It is not the most efficient sorting algorithm for large datasets, but it is simple and effective for the moderate dataset sizes in this project, ensuring correct selection of k-nearest neighbours.

Euclidean Distance

Used as the metric to calculate the similarity between data points. It determines the straight-line distance between two points in multi-dimensional space, preserving spatial relationships

within the dataset. This choice is particularly suited for numerical data, ensuring that differences in feature values are appropriately captured during classification. The Euclidean distance formula is: $d(a, b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$

Comparisons

In this project, the comparison of the accuracy among the three implemented machine learning models (SVM, FNN, and KNN) helps to explain the strengths and suitability of each approach for the classification task.

SVM with an accuracy of 95.69%, it is and effective approach in high-dimensional spaces through the use of the RBF kernel, Euclidean distance metric, and one-vs-one strategy. It handles complex decision boundaries, it is reliable when handling datasets where the classes are not linearly separable, and it's the most efficient option in terms of computational cost while still achieving good accuracy. FNN with an accuracy of 96.60%, it is capable to learn intricate patterns through backward propagation and the hidden layer design makes it flexible. It's computational efficiency was moderate but almost double to the one achieved by the SVM. This approach achieves a considered balance between speed and accuracy. KNN with an accuracy of 98.29%, it captures local data relationships using Euclidean distance metric, and majority voting among neighbours. Although is the slowest model to produce results due to requiring the computation of distances to all training points for every test instance. Providing the best accuracy but with a high computational cost.

Conclusion

By successfully implementing and evaluating three machine learning models(SVM, FNN, and KNN) to classify handwritten digits from the Optical Recognition of Handwritten Digits datasets. The comparison of these models highlighted the balance between accuracy, computational efficiency, and suitability for different classification tasks.

Normalisation and one-hot encoding of the data enhanced the learning process by providing the model with appropriately scaled inputs and a clear, mathematically consistent way to understand target categories. Hyperparameter tuning was another important step to optimise the model and ensuring that the selected configurations lead to robust and high-performing systems. Using K-Fold Cross-Validation in each machine learning model, helped figuring out how the model is performing during the testing with new data. At last, the confusion matrices and derived performance metrics usage comprehensively evaluated the machine learning models, helping to identify areas for improvement and supporting the development of more reliable and effective classifications.

The Support Vector Machines (SVM) with the Radial Basis Function (RBF) kernel and one-vs-one classification strategy demonstrated a low computational cost and efficient approach. By decomposing the task into multiple binary classifications, the one-vs-one approach ensured that each pair of classes is individually distinguished, enabling precise decision-making through a voting mechanism. Filtering the data for each binary classifier simplified the optimisation process, reducing computational complexity and focusing on the specific distinctions between two labels. The use of the RBF kernel enhanced the SVM's capability to handle non-linear class boundaries by projecting the data into higher-dimensional spaces, capturing intricate patterns effectively. Mathematical principles such as Lagrange multipliers,

Sequential Minimal Optimisation (SMO), and Karush-Kuhn-Tucker (KKT) conditions were a key to the SVM's design, ensuring that the decision boundary maximised the margin between classes while balancing misclassification penalties.

The Feedforward Neural Network (FNN) also demonstrated an efficient approach with a moderate computational cost. By using the ReLU activation function as support in the hidden layer, the network addressed the vanishing gradient problem, ensuring efficient gradient flow and stable learning of complex relationships in the data. The Softmax function in the output layer enabled the model to produce well-calibrated probabilities, making predictions interpretable and suitable for multi-class scenarios. Through iterative weight optimisation via forward and backward propagation, combined with cross-entropy loss minimisation, the FNN effectively learned to align its predictions with true class labels.

The K-Nearest Neighbour (KNN) algorithm it's simple and effective as a classification method. By leveraging the Euclidean distance metric, the algorithm captured the spatial relationships between data points. The use of the "Most Common Label" approach ensured accurate predictions through majority voting among the k-nearest neighbors, while the Bubble Sort algorithm effectively identified these neighbours by sorting distances. Although Bubble Sort is not the most efficient for larger datasets, it performed well within the moderate dataset sizes of this project, maintaining clarity and correctness in the implementation.

While KNN excelled in accuracy with 98.29%, its computational inefficiency may limit its practicality for larger datasets. In cases of time-sensitive tasks, SVM reliable performance, would still achieve a strong accuracy of 95.69%. FNN also provided a balanced performance with 96.60% offering a second alternative for situations in which accuracy and computational efficiency are in consideration. For this classification task the best choice would be KNN due to the main priority being high accuracy. Although if the priority was a balance between low computational cost and achieving high accuracy, the best option would be the use of SVM models, which with further improvement could possibly reach higher accuracy. FNN could be an alternative, but it's computational cost it is not as low as the one achieved by the SVM, and its accuracy is not extremely higher than the SVM to make it a more suitable option.

Self-Marking Table For The Assignment

For the self-assessment of this project, considering the implementation of multiple machine learning models to demonstrate knowledge in each one of the models and additionally demonstrating the importance of selecting the right algorithm for different tasks, the grades achieved are reflected in the following table:

To Evaluate	Grade Achieved	Reason
Self-Marking Sheet	10 **out of** 10	Inclusion of a self-marking table in the report.
Running Code	10 **out of** 10	Running code without errors.
Two-Fold Test	5 **out of** 5	Two-Fold Test implemented and versatile for different machine learning models
Quality of Code	15 **out of** 15	Code structure is modular and versatile for different machine learning models with good size functions, reasonably explaining the importance of the algorithms and

		approach in the comments, with good use of variables, and named constants.
Report	18 **out of** 20	Report is brief on the explanation of the machine learning algorithms used to increase understanding in machine learning and generate comparisons.
Quality of Results	19 **out of** 20	The quality of the results were good in all the different models.
Quality of Algorithm	19 **out of** 20	The algorithms implemented were modular and clear with a good accuracy achieved and computational cost.

Annexes

```
================================================================================================================
|                  Evaluating all hyperparameters combinations for the best accuracy with SVM                  |
================================================================================================================

Best accuracy achieved...
================================================================================================================
|                              MACHINE LEARNING --- Support Vector Machine (SVM)                               |
================================================================================================================
|                                    K-Fold Cross-Validation --- 2 - Fold                                      |
================================================================================================================
| Parameters                                                                                                     |
| Values = 64 | Labels = 10 | Gamma = 100.000 | C = 1.000 | Maximum Iterations = 10.00 | Tolerance = 0.00001     |
| Total accuracy = 95.6940%                                                                                      |
================================================================================================================
|                                            Confusion Matrix                                                    |
================================================================================================================
|            | Pred 0  | Pred 1  | Pred 2  | Pred 3  | Pred 4  | Pred 5  | Pred 6  | Pred 7  | Pred 8  | Pred 9  |
----------------------------------------------------------------------------------------------------------------
|  Actual  0 |     550 |       1 |       0 |       0 |       3 |       0 |       0 |       0 |       0 |       0 |
|  Actual  1 |       0 |     564 |       0 |       1 |       1 |       0 |       1 |       2 |       0 |       2 |
|  Actual  2 |       0 |      10 |     531 |       9 |       0 |       0 |       1 |       3 |       2 |       1 |
|  Actual  3 |       0 |       3 |       1 |     544 |       0 |       7 |       0 |       5 |       6 |       6 |
|  Actual  4 |       0 |       1 |       0 |       0 |     556 |       0 |       4 |       1 |       3 |       3 |
|  Actual  5 |       1 |       0 |       0 |       1 |       0 |     550 |       0 |       0 |       0 |       6 |
|  Actual  6 |       5 |       3 |       0 |       0 |       1 |       1 |     546 |       0 |       2 |       0 |
|  Actual  7 |       0 |       1 |       0 |       1 |       7 |       0 |       0 |     543 |       0 |      14 |
|  Actual  8 |       0 |      40 |       0 |       2 |       2 |       0 |       3 |       1 |     501 |       5 |
|  Actual  9 |       0 |       5 |       0 |       8 |      10 |       9 |       0 |      30 |       7 |     493 |
================================================================================================================
| Label             | Precision                 | Recall                    | F1-Score                  |
================================================================================================================
| Label 0           |                98.9209% |                99.2780% |                99.0991% |
| Label 1           |                89.8089% |                98.7741% |                94.0784% |
| Label 2           |                99.8120% |                95.3321% |                97.5207% |
| Label 3           |                96.1131% |                95.1049% |                95.6063% |
| Label 4           |                95.8621% |                97.8873% |                96.8641% |
| Label 5           |                97.0018% |                98.5663% |                97.7778% |
| Label 6           |                98.3784% |                97.8495% |                98.1132% |
| Label 7           |                92.8205% |                95.9364% |                94.3527% |
| Label 8           |                96.1612% |                90.4332% |                93.2093% |
| Label 9           |                93.0189% |                87.7224% |                90.2930% |
================================================================================================================
```

Fig. 1 SVM - Best Accuracy Achieved

```
Evaluating all hyperparameters combinations for the best accuracy with FNN

Best accuracy achieved...

MACHINE LEARNING --- Feedforward Neural Network (FNN)

K-Fold Cross-Validation --- 2 - Fold

Parameters
Input Neurons = 64 | Output Neurons = 10 | Hidden Neurons = 250.00 | Learning Rate = 0.050 | Epochs = 750.00
Total accuracy = 96.6014%

Confusion Matrix
```

	Pred 0	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5	Pred 6	Pred 7	Pred 8	Pred 9
Actual 0	548	0	2	0	2	1	1	0	0	0
Actual 1	1	559	1	0	1	0	1	2	6	0
Actual 2	1	4	547	2	0	0	1	0	2	0
Actual 3	1	0	3	555	0	8	0	0	1	4
Actual 4	2	3	0	0	549	0	6	1	3	4
Actual 5	0	0	4	6	1	537	1	0	1	8
Actual 6	3	2	1	0	5	0	546	0	1	0
Actual 7	0	0	0	1	2	8	0	546	2	7
Actual 8	1	14	1	2	5	5	0	0	515	11
Actual 9	3	5	0	3	6	5	0	10	3	527

Label	Precision	Recall	F1-Score
Label 0	97.8571%	98.9170%	98.3842%
Label 1	95.2300%	97.8984%	96.5458%
Label 2	97.8533%	98.2047%	98.0287%
Label 3	97.5395%	97.0280%	97.2831%
Label 4	96.1471%	96.6549%	96.4004%
Label 5	95.2128%	96.2366%	95.7219%
Label 6	98.2014%	97.8495%	98.0251%
Label 7	97.6744%	96.4664%	97.0667%
Label 8	96.4419%	92.9603%	94.6691%
Label 9	93.9394%	93.7722%	93.8557%

Fig. 2 FNN - Best Accuracy Achieved

```
Evaluating all hyperparameters combinations for the best accuracy with KNN

Best accuracy achieved...

MACHINE LEARNING --- K-Nearest Neighbour (KNN)

K-Fold Cross-Validation --- 2 - Fold

Parameters
Values = 64 | Labels = 10 | K value = 4.00
Total accuracy = 98.2918%

Confusion Matrix
```

	Pred 0	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5	Pred 6	Pred 7	Pred 8	Pred 9
Actual 0	553	0	0	0	0	0	1	0	0	0
Actual 1	0	567	0	0	0	0	1	1	0	2
Actual 2	0	1	555	0	0	0	0	1	0	0
Actual 3	0	2	0	559	0	2	0	1	3	5
Actual 4	1	2	0	0	554	0	2	1	1	7
Actual 5	0	0	0	2	1	546	0	0	1	8
Actual 6	0	2	0	0	0	0	556	0	0	0
Actual 7	0	0	0	1	1	0	0	560	1	3
Actual 8	0	19	0	2	0	0	0	0	531	2
Actual 9	0	2	0	8	1	1	0	4	3	543

Label	Precision	Recall	F1-Score
Label 0	99.8195%	99.8195%	99.8195%
Label 1	95.2941%	99.2995%	97.2556%
Label 2	100.0000%	99.6409%	99.8201%
Label 3	97.7273%	97.7273%	97.7273%
Label 4	99.4614%	97.5352%	98.4889%
Label 5	99.4536%	97.8495%	98.6450%
Label 6	99.2857%	99.6416%	99.4633%
Label 7	98.5915%	98.9399%	98.7654%
Label 8	98.3333%	95.8484%	97.0750%
Label 9	95.2632%	96.6192%	95.9364%

Fig. 3 KNN - Best Accuracy Achieved

References

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Villegas, F., (n.d). ‘Data manipulation: What it is, Techniques & Examples’. *QuestionPro.* Available at: https://www.questionpro.com/blog/data-manipulation/ (Accessed: 2 December 2024).

Kumari, J., (2024). 'Understanding parameters and hyperparameters', *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2024/06/parameters-and-hyperparameters/ (Accessed: 2 December 2024).

Shah, E., (2024). ‘K Fold Cross Validation in Machine Learning’. *Scaler.* Available at: https://www.scaler.com/topics/k-fold-cross-validation-in-machine-learning/ (Accessed: 2 December 2024).

IBM, (2023). ‘What is a confusion matrix?’, *IBM*. Available at: https://www.ibm.com/topics/confusion-matrix (Accessed: 2 December 2024).

GeeksforGeeks, (2023). ‘F1-score in machine learning’, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/f1-score-in-machine-learning/ (Accessed: 2 December 2024).

GeeksforGeeks, (2024). 'K-Nearest Neighbour algorithm', *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/k-nearest-neighbors/ (Accessed: 5 December 2024).