Supervised Learning: Assignment 3

Natalie Alexander

1. Abstract

In this assignment, neural net and support vector machine models were built to classify handwritten images as even or odd numbers. Parameter tuning was performed to obtain the optimal model. Finally, the best model (a neural net model with test accuracy = 0.950), was used to classify an unseen, unlabeled data set.

2. Introduction

Support vector machines (SVMs) and neural networks (NNs) are used for classification of categorical- and regression of numeric-dependent variables. The SVM algorithm works by generating a decision boundary or hyperplane that separates the dependent variable into distinct binary classes. A kernel function is used to transform the data into a high-dimensional feature space. As a result, the non-linear boundary of the untransformed feature space becomes a linear boundary in the transformed feature space. The Gaussian kernel function, which is implemented in this assignment, is used when there is no prior knowledge about the data [1]. On the other hand, neural networks are powerful and versatile when it comes to learning complex relationships and patterns in high-dimensional data. Neural networks are not limited to binary outcomes and are the preferred model choice in image recognition [2].

3. Exploratory data analysis

i. Data set

The train_new data set has 37801 observations and 784 independent variables ("pixel0" to "pixel783"). A sample of the training data is provided in **table 1 and 2**. Further analysis reveals that the minimum pixel value is 0 and the maximum pixel value is 255. The pixel values represent the intensity of the colour. Here 0 represents black and 255 represents white. The categorical dependent variable ('label') was then converted to a factor. As seen before in **tables 1 and 2**, the dependent variable 'label' has two categories: odd and even.

Table 1: Head of training data. The 'label' column represents the dependent variable and the 'pixel' columns represent the independent variables. The first 3 observations and independent variables are shown.

label	pixel0	pixel1	pixel2
odd	0	0	0
even	0	0	0
odd	0	0	0

ii. Visualization

Table 2: Tail of training data. The 'label' column represents the dependent variable and the 'pixel' columns represent the independent variables. The last 3 observations and independent variables are shown.

	label	pixel781	pixel782	pixel783
37799	odd	0	0	0
37800	even	0	0	0
37801	odd	0	0	0

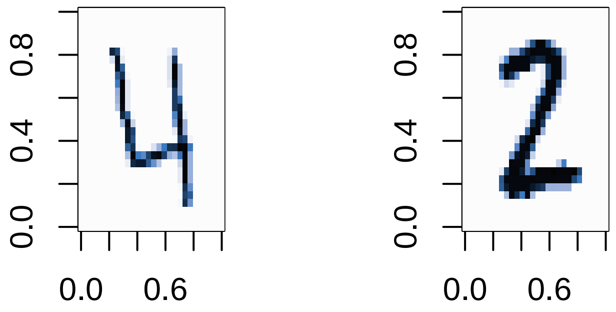

Figure 1: Images of pixel data. Left: Observation 4. Right: Observation 22.

In **figure 1**, observation 4 (left) and observation 22 (right) produced the numbers 4 and 2, respectively. In order to generate the images, the independent variables (pixel 0 to pixel 783) were converted to a 28 x 28 matrix. The images were then rotated 90 degrees in a clockwise direction.

iii. Distribution and proportions

The dependent variable of the training set has 50.996% 'odd' and 49.004% 'even' observations. We observe approximate class balance. The histograms in **figure 2**, show the frequencies of each pixel distributed across the odd class (left) and the even class (right). For both odd and even classes, we see a similar distribution where pixel 0 and pixel 783 are the highest occurring pixels, while the central pixels occur at much lower frequencies. In both classes we see a bimodal distribution.

4. Methods

4.1 Neural net model construction

i. Overview

Pixel data ranges from 0 to 255, however many sources recommend scaling pixel values to range between 0-1 [3-5]. As recommended, all independent variables (pixels) in the train_new data set were divided by 255. The scaled data set was then split into 80% training- and 20% test sets, using the caret package. By default the caret package ensures stratified sampling. Using the h2o package, both training and test sets were converted to h2o format. Ten-fold cross-validation was implemented to build each neural-net model on the training set. Model parameters were chosen in the direction of the reduced mean cross-validation log-loss.

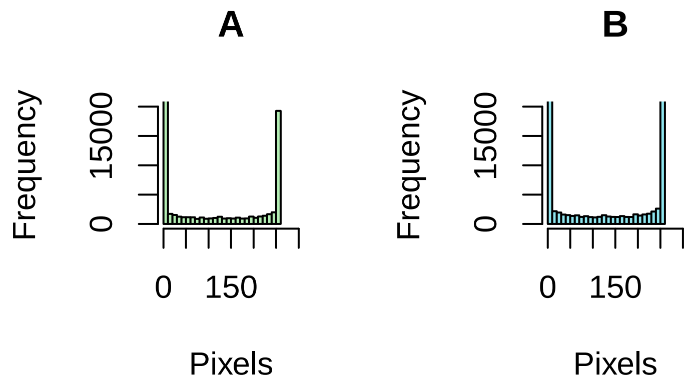

Figure 2: Histograms showing the frequency of each pixel value for each class of the dependent variable. A) Odd numbers. B) Even numbers.

The optimal activation function, number of hidden layers, number of hidden neurons, and the learning rate were determined by the model with the minimum cross-validation (CV) log-loss. Subsequently, L1 (LASSO) and L2 (ridge) regularization was performed on the training models. Input and drop-out methods were also implemented.

ii. Activation function

Parameters of the neural net models can be seen in **table 3** and the CV log-loss can be seen in **figure 3**. The initial model (nn1) used a common activation function (rectifier). However, the CV log-loss of 0.585 was indicative of poor model performance on the training set. Training model nn2 then used a tanh activation function, which decreased the CV log-loss to 0.260. As a result, the tanh activation function was used in subsequent models. The tanh activation function is an excellent alternative to the sigmoid activation function. The tanh activation function enables rapid learning and convergence. In addition, the tanh activation function is commonly used in binary classification problems [6-7].

iii. Hidden layers and neurons

A range of hidden layers(hidden neurons) were assessed, from 1(2) in models nn1 to nn2 and 1(3), 1(4) and 2(2, 1), in models nn3 to nn5, respectively. Fewer hidden layers(hidden neurons) served as a proxy for simpler models, while a larger number of hidden layers(hidden neurons) served as a proxy for more complex models. If there was no improvement in the CV log-loss, further increase or decrease in the number of hidden layers (hidden neurons) was not required. The CV log-loss decreased as the number of hidden neurons increased (in a single hidden layer). However, the CV log-loss increased as the number of overall hidden layers increased. Model nn4 with 1 hidden layer (4 hidden neurons) had a minimum CV log-loss of 0.203 and was selected as the best model going forward (**figure 3**).

iv. Learning rate (epsilon)

The learning rate was then assessed, increasing and decreasing the learning rate relative to the default learning rate of 1e-08 in model nn4. Learning rates of 1e-05 and 1e-011 were used in models nn6 and nn7, respectively. We expect faster convergence with a larger learning rate, relative to the smaller learning rate. However, both models nn6 and nn7 performed poorly in comparison to model nn4. Thus, model nn4 remained the best model (**figure 3**).

v. L1 vs L2 regularization

L1 and L2 regularization was then performed to reduce over-fitting of the training model, simplifying the training model as a result. The regularization parameter was consistent with current pipelines which either gradually increase or decrease their lambda values by a factor of 10 [6]. L1 regularization was used in models nn8 to nn12 to shrink weights towards 0 and also drop these insignificant independent variables. Feature selection was performed as a result. The regularization parameter ranged from 1e-1 to 1e-5 (**table 3**). Model nn8 with a regularization parameter of 1e-3, had the best performance and reduced the CV log-loss from 0.203 (model nn4) to 0.150 (model nn8). Model nn8 was the best model going forward. L2 regularization was also assessed to reduce the weights in models nn13 to nn17, with regularization parameters ranging from 1e-1 to 1e-5, respectively. However, model nn8 still outperformed the L2 regularized models in terms of CV log-loss (**figure 3**).

vi. Input drop-out ratio

The input drop-out ratio was tuned to improve model generalization and to prevent model learning and reliance on specific input neurons [9]. The input drop-out ratio ensures model robustness to diverse representations of unseen, test data. Until this point, an input drop-out ratio of 0.1 was used, in accordance with recommendations by the h2o documentation. Model nn18 is a modified version of model nn8, with an input drop-out ratio of 0.2, as recommendation by the h2o documentation. However, model nn8 still had the minimum CV log-loss (0.150) relative to model nn18 with a CV log-loss of 0.155 (**figure 3**).

vii. Hidden layer drop-out ratios

Hidden layer dropout ratios were then tuned to improve model generalization and to prevent co-adaptation of hidden layer neurons [9]. Models nn19 to nn23 had hidden layer drop-out ratios of 0.1 to 0.5, respectively. However, model nn8 still had the minimum CV log-loss, and was chosen as the best neural net model (**figure 3**).

ix. Predictions on test data

As a result of normalization on the training and validation sets, predictions were made on a scaled test set, where pixel values were also divided by 255. Model nn8 with parameters shown in **table 3** was used for test set prediction. Labels were assigned to the test set based on the class (even or odd) with the largest predicted probability.

Table 3: Neural net models built with ten-fold cross-validation

Model ID	Activation function	Hidden layers(neurons)	Epsilon	Epochs	Input drop-out ratio	Hidden drop-out ratios	L1 regularization	L2 regularization	CV log-loss
nn1	Rectifier	1(2)	1e-08	1000	0.1	0.0	0.0	0.0	0.585
nn2	tanh	1(2)	1e-08	1000	0.1	0.0	0.0	0.0	0.260
nn3	tanh	1(3)	1e-08	1000	0.1	0.0	0.0	0.0	0.215
nn4	tanh	1(4)	1e-08	1000	0.1	0.0	0.0	0.0	0.203
nn5	tanh	2(2, 1)	1e-08	1000	0.1	0.0	0.0	0.0	0.252
nn6	tanh	1(4)	1e-05	1000	0.1	0.0	0.0	0.0	0.226
nn7	tanh	1(4)	1e-011	1000	0.1	0.0	0.0	0.0	0.206
nn8	tanh	1(4)	1e-08	1000	0.1	0.0	0.001	0.0	0.150
nn9	tanh	1(4)	1e-08	1000	0.1	0.0	1e-05	0.0	0.192
nn10	tanh	1(4)	1e-08	1000	0.1	0.0	0.1	0.0	0.529
nn11	tanh	1(4)	1e-08	1000	0.1	0.0	0.01	0.0	0.290
nn12	tanh	1(4)	1e-08	1000	0.1	0.0	1e-04	0.0	0.164
nn13	tanh	1(4)	1e-08	1000	0.1	0.0	0.0	0.1	0.298
nn14	tanh	1(4)	1e-08	1000	0.1	0.0	0.0	0.01	0.188
nn15	tanh	1(4)	1e-08	1000	0.1	0.0	0.0	0.001	0.183
nn16	tanh	1(4)	1e-08	1000	0.1	0.0	0.0	1e-04	0.201
nn17	tanh	1(4)	1e-08	1000	0.1	0.0	0.0	1e-05	0.201
nn18	tanh	1(4)	1e-08	1000	0.2	0.0	0.001	0.0	0.155
nn19	tanh	1(4)	1e-08	1000	0.1	0.1	0.001	0.0	0.238
nn20	tanh	1(4)	1e-08	1000	0.1	0.2	0.001	0.0	0.301
nn21	tanh	1(4)	1e-08	1000	0.1	0.3	0.001	0.0	0.353
nn22	tanh	1(4)	1e-08	1000	0.1	0.4	0.001	0.0	0.383
nn23	tanh	1(4)	1e-08	1000	0.1	0.5	0.001	0.0	0.355

[1] Number of independent variables = 784. Number of dependent variables = 1. Number of dependent variable categories = 2.
[2] Train size = 27218. Validation size = 3024. Test size = 7559

4.2 Support Vector Machine (SVM) model construction

i. Overview

For SVM model construction, h2o's PSVM function was implemented, which only solves binary classification problems. The entire train_new data set was scaled by dividing the independent variables (pixels) by 255.

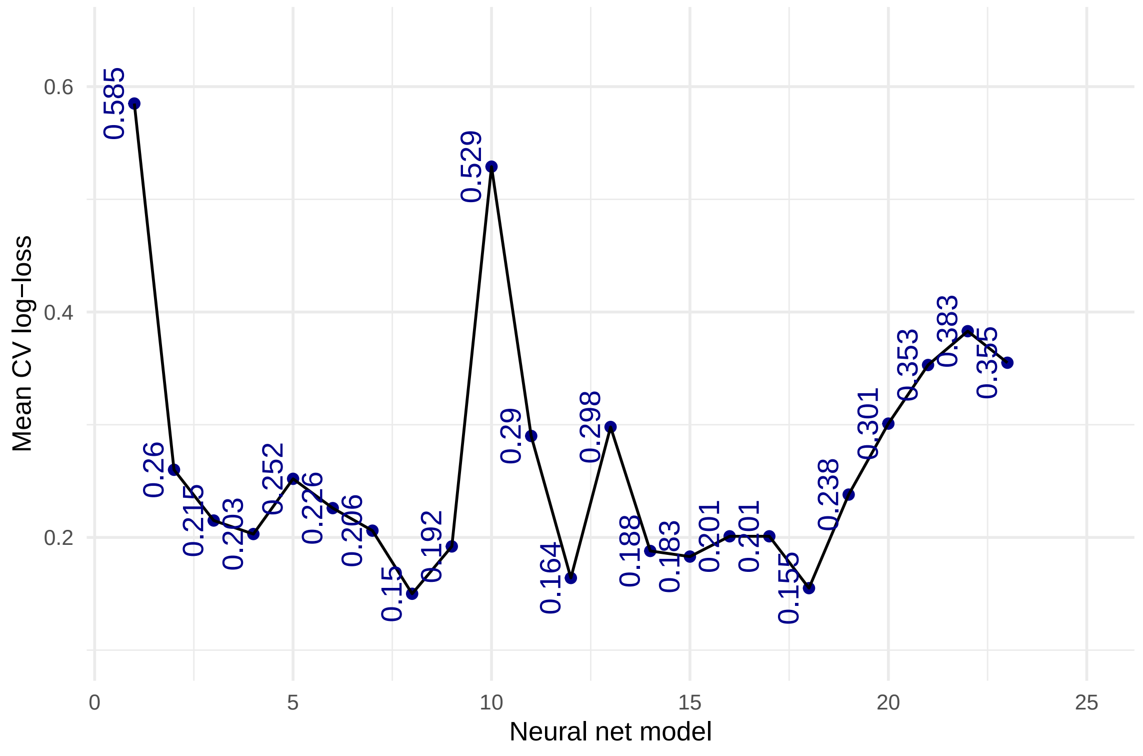

Figure 3: Mean ten-fold cross-validation log-loss across neural net models

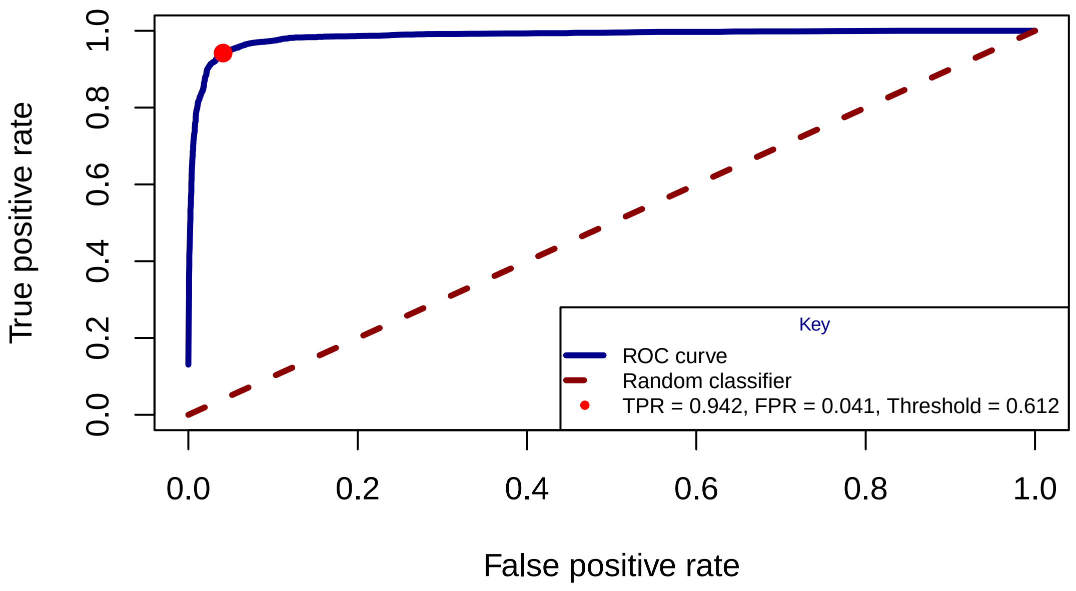

Figure 4: Test set ROC curve for neural net model nn8

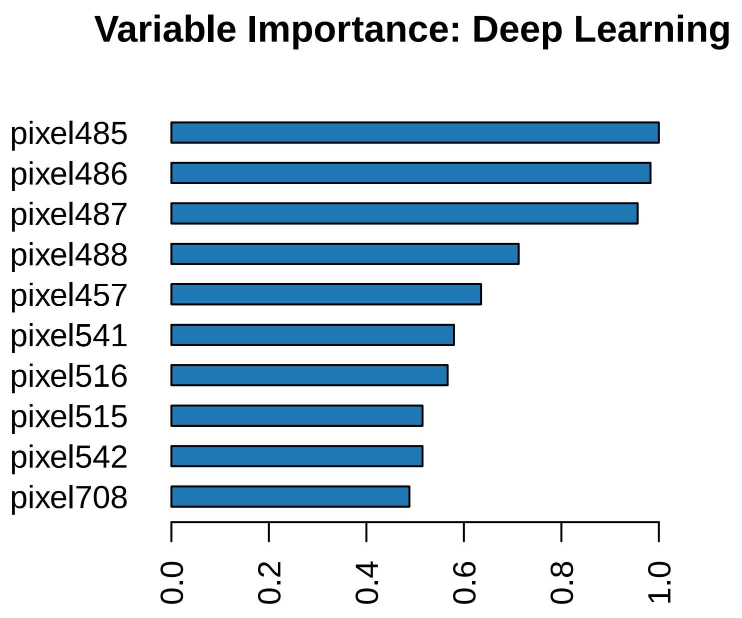

Figure 5: Variable importance plot of neural net model nn8. The top pixels represent a large reduction in squared training error. The bottom pixels represent a smaller reduction in squared training error

In this way, the independent variables were normalized to range between 0 and 1. The scaled data was then split into 80% training and 20% test sets. The training data was further split into 80% training data and 20% validation data sets. All data sets were converted to h2o format. The models were built on the new training set, while the validation set was used to calculate the mean misclassification rate. The model with the minimum mean misclassification rate on the validation set, was the optimal model. The best SVM model could then be used to test model performance on the test set.

ii. Kernel and cost

Table 4 shows the SVM model parameters and **figure 6** shows the mean misclassification rate on the validation set. H2o's PSVM only supports Gaussian kernel functions, and so a Gaussian kernel function was applied to all models. The cost value, also known as the regularization parameter, was tuned in the direction of reducing mean misclassification rate on the validation set. The cost parameter was tuned using C = 1, 0.1, 0.01 and 3 in models svm1 to svm4, respectively. The cost parameter served as a regularization mechanism [10]. Increasing the cost value applies a larger penalty to model misclassification, reducing the margin and number of support vectors. Decreasing cost values applies a smaller penalty to model misclassification, increasing the margin and number of support vectors. Notably, the number of support vectors did not decrease across the models (**table 4**). Model svm2 had the minimum mean misclassification rate (0.198, **figure 6**) on the validation set and so cost = 0.1 was chosen as the cost parameter going forward.

iii. Gamma

Thereafter, the gamma parameter was tuned to minimize the mean misclassification rate on the validation set, using gamma = 1, 0.3 and 0.5 in models svm5 to svm7, respectively. The gamma value controls the radius of observations that influence the decision boundary [10]. I chose large gamma values as a proxy for far-away observations that influence the decision boundary. I also chose small gamma values as a proxy for close-by observations that influence the decision boundary. However, model svm2 with default gamma = -1 still had the best mean misclassification rate (0.198) on the validation set (**figure 6**).

iv. Predictions on the test set

Overall, model svm2, with cost = 0.1 and gamma = -1, had the minimum mean misclassification rate (0.198) on the validation set (**figure 6**). **Table 4** shows all optimal parameters for model svm2. As a result of normalization on the training and validation sets, predictions were made on a scaled test set, where pixel values were divided by 255. Model svm2 was used to make predictions on the test set. Labels were assigned to the scaled test set according to the class (even or odd) with the largest predicted probability.

Table 4: SVM models built using the hold-out method

Model ID	Kernel	Cost	Gamma	No. support vectors	Mean validation misclassification rate
svm1	Gaussian	1.00	-1.0	24194	0.201
svm2	Gaussian	0.10	-1.0	24194	0.198
svm3	Gaussian	0.01	-1.0	24194	0.219
svm4	Gaussian	3.00	-1.0	24194	0.201
svm5	Gaussian	0.10	1.0	24194	0.510
svm6	Gaussian	0.10	0.3	24194	0.436
svm7	Gaussian	0.10	0.5	24194	0.496

[1] Number of independent variables = 784. Number of dependent variables = 1. Number of dependent variable categories = 2

[2] Train size = 24194. Validation size = 6048. Test size = 7559

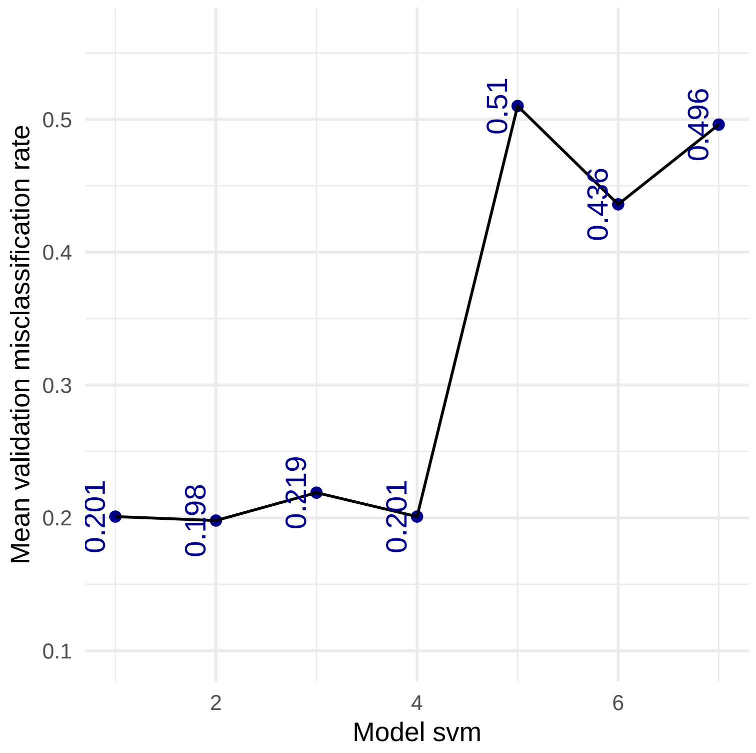

Figure 6: Mean misclassification rate on the validation set across all SVM models

4.3. Unlabeled data

The unlabeled test_new data set was scaled, by dividing pixel values by 255. The best model, neural net model nn8, was used to make predictions on the scaled test_new data set. For comparison purposes, the best SVM model (svm2) was also used to make predictions on the scaled test_new data. Labels for each observation was determined by assuming the class with the largest predicted probability.

5. Results and Discussion

5.1. Model performance and comparison

i. Overview

Table 5 shows that model nn8 had an overall better model performance on both the training and test sets, relative to model svm2. The performance metrics considers the 'even' and 'odd' label as the positive and negative class, respectively.

ii. Accuracy

Both models had relatively good performance on the training set. However, The neural net model nn8 had a training accuracy of 0.953 which is larger than the 0.808 accuracy obtained by the support vector machine model svm2. Model nn8 had a test accuracy of 0.950, with an insubstantial reduction in accuracy relative to its training accuracy of 0.953. Model svm2 had a test accuracy of 0.808, with no improvement relative to its training accuracy of 0.808. Model nn8's test accuracy was also larger than that of model svm2's test accuracy.

iii. Recall

Model nn8 had a recall of 0.954 on the test set, suggesting that 95.4% of positive observations (even) were correctly classified as true positives (even). Model svm2 had a recall of 0.765 on the test set, suggesting that 76.5% of positive observations (even) were correctly classified as true positives (even). Model nn8 had a 18.9% larger recall on the test set relative to model svm2, when considering recall as a percentage. Model nn8 is thus better at classifying true positives (even cases).

iv. Specificity

Model nn8 had a specificity of 0.946 on the test set which is larger than model svm2 with a specificity of 0.849 on the test set. For model nn8, 94.6% of negative (odd) observations were correctly classified as true negatives (odd). For model svm2, 84.9% of its negative (odd) observations were correctly classified as true negatives (odd). Model nn8 had a 9.7% larger specificity on the test set than model svm2, when considering specificity as a percentage. Model nn8 is thus better at classifying true negatives (odd cases).

v. Precision

Model nn8 had a precision value of 0.944 on the test set, suggesting 94.4% of positive predictions (even) were actual true positives (even). Model svm2 had a precision value of 0.829 on the test set, suggesting 82.9% of positive predictions (even) were actual true positives (even). Model nn8 had a 11.5% larger precision relative to model svm2, when considering precision as a percentage.

vi. F1-score

Model nn8 had an F1-score of 0.954 on the test set, suggesting that the model's ability to capture positive cases (while keeping false positives and false negatives low) is very good. Model svm2 had a F1-score of 0.765 on the test set, suggesting the model's ability to capture positive cases (while keeping false positives and false negatives low) is okay. Model nn8 had a 18.9% larger test F1-score relative to model svm2, when considering the F1-score as a percentage.

vii. Matthew's correlation coefficient

Finally, Matthew's correlation coefficient (MCC) was assessed, which summarizes the metrics in the confusion matrix (true positives, true negatives, false positives and false negatives) into a single metric. Model nn8 had a MCC of 0.9, which is close to 1, suggesting almost perfect agreement between the predictions and the actual classes of the observations. Model svm2 had a MCC of 0.616, which suggests that the model does not do the best job at classifying true cases. There is some disagreement between the predictions and the actual classes of the observations. Model nn8 had a 28.4% larger MCC on the test set relative to model svm2, when considering MCC as a percentage.

Table 5: Model performance and comparison

	Model 1	Model 2
Model ID	nn8	svm2
Model type	Neural network	Support vector machine
Train size	27218 (9 folds)	24194
Test size	7559	7599
Validation size	3024 (1 fold)	6048
Activation function	tanh	NA
Hidden layers(neurons)	1(4)	NA
Learning rate	1e-08	NA
Epochs	1000	NA
Input drop-out ratio	0.1	NA
Hidden drop-out ratios	0.0	NA
L1 regularization	0.001	NA
L2 regularization	0.0	NA
Cross-validation	10-fold	NA
Hold-out	NA	Yes
Kernel	NA	Gaussian
Cost	NA	0.1
Gamma	NA	-1
Train accuracy	0.953	0.808
Test accuracy	0.95	0.808
Test recall	0.954	0.765
Test specificity	0.946	0.849
Test Precision	0.944	0.829
Test F1-score	0.954	0.765
Test MCC	0.9	0.616

[1] Positive class : even

[2] Negative class : odd

[3] NA: not applicable

5.2. Best model evaluation (neural net nn8)

i. Variable importance

In the variable importance plot (**figure 5**), we see that only 10 independent variables had a significant contribution to the training model. Pixel 485 (top) had the most significant predictive power, with the largest reduction in the squared training error. In contrast, pixel 708 (bottom) had the least predictive power with the smallest reduction in the squared training error, among the 10 independent variables.

ii. Weights and biases

Table 6 shows the weights of the input layer to the hidden layer, which ranges from -0.331 to 0.162. Thus, the input neurons have a small positive and sometimes negative contribution to the hidden layer's activation function and output. **Table 7** shows the weights of the hidden layer to the output layer which ranges from -3.139 to 2.986, suggesting a larger positive and sometimes negative contribution of the hidden layer neurons to the output of the activation function in the output layer. **Table 8** shows the bias of the input layer to the neurons in the hidden layer, where the bias ranges from -1.374 to 1.55, suggesting moderate-positive and sometimes negative contribution of the bias to the output of the activation function in the hidden layer. **Table 8** also shows the bias of the hidden layer to the neurons in the output layer, where the bias ranges from -1.843 to 1.836, suggesting a larger positive and sometimes negative contribution of the bias to the output of the activation function in the output layer.

Table 6: Weights of input layer (columns) to hidden layer (rows) in neural net model nn8

	pixel485	pixel486	pixel487	pixel488	pixel457	pixel541	pixel516	pixel515	pixel542	pixel708
Hidden neuron 1	-0.007	-0.005	0.004	0.002	-0.019	0.061	-0.008	0.001	0.007	0.000
Hidden neuron 2	-0.331	-0.320	-0.300	-0.226	-0.198	0.079	-0.094	-0.110	0.049	0.162
Hidden neuron 3	-0.013	-0.030	-0.029	-0.008	0.012	0.020	-0.083	-0.059	0.017	-0.002
Hidden neuron 4	0.005	0.002	0.033	0.036	0.024	-0.264	-0.168	-0.109	-0.327	0.010

[1] Weights of variable important pixels

Table 7: Weights of hidden layer (columns) to output layer (rows) in neural net model nn8

	Hidden neuron 1	Hidden neuron 2	Hidden neuron 3	Hidden neuron 4
Even	-0.159	-2.687	-0.175	-0.019
Odd	1.275	0.332	-3.139	2.986

Table 8: Bias of neural network model nn8

	Bias
Input layer to hidden neuron 1	0.415
Input layer to hidden neuron 2	-1.374
Input layer to hidden neuron 3	1.550
Input layer to hidden neuron 4	-1.233
Hidden layer to output(even)	-1.843
Hidden layer to output (odd)	1.836

iii. ROC curve

Model nn8 has a ROC AUC of 0.988, 0.985 and 0.986 for its training, cross-validation and test sets, respectively. There is a marginal decrease in the ROC AUC from training to test and cross-validation sets. The ROC AUC metrics are close to 1, suggesting that model nn8 is close to a perfect classifier. The test set ROC AUC of 0.986, suggests that model nn8 has excellent discriminatory power between positive and negative cases, and generalizes well to unseen data. In addition, **figure 4** shows that the ROC curve is close to the top-left corner, indicative of a large true positive rate (TPR) and small false positive rate (FPR). The best classifier has a TPR = 0.942 and FPR = 0.041 at threshold = 0.612.

6. Conclusion

In conclusion, neural net models are far more powerful than SVMs in terms of model performance on both the training and test sets. In addition, SVMs are computationally infeasible. Neural net models are the preferred choice in terms of algorithm speed, computational feasibility, model complexity and accuracy.

References

1. Team, D. (2021) Kernel functions-introduction to SVM Kernel andamp; Examples, DataFlair. Available at: https://data-flair.training/blogs/svm-kernel-functions/ (Accessed: 03 June 2023).

2. Editor (2020) Image recognition with deep neural networks and its use cases, AltexSoft. Available at: https://www.altexsoft.com/blog/image-recognition-neural-networks-use-cases/ (Accessed: 03 June 2023).

3. Brownlee, J. (2019) How to manually scale image pixel data for deep learning, MachineLearningMastery.com. Available at: https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/ (Accessed: 01 June 2023).

4. How to normalize, center, and standardize image pixels in Keras? (2023) GeeksforGeeks. Available at: https://www.geeksforgeeks.org/how-to-normalize-center-and-standardize-image-pixels-in-keras/ (Accessed: 01 June 2023).

5. Murillo-Escobar, M.A., Cruz-Hernández, C., Abundiz-Pérez, F., López-Gutiérrez, R.M. and Del Campo, O.A., 2015. A RGB image encryption algorithm based on total plain image characteristics and chaos. Signal Processing, 109, pp.119-131.

6. Sharma, S. (2022) Activation functions in neural networks, Medium. Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (Accessed: 03 June 2023).

7. TanhExp: A smooth activation function - wiley online library. Available at: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/cvi2.12020 (Accessed: 03 June 2023).

8. Kumar, G.S. (2020) Regularization: Machine learning, Medium. Available at: https://towardsdatascience.com/regularization-machine-learning-891e9a62c58d (Accessed: 03 June 2023).

9. Brownlee, J. (2022) Dropout regularization in deep learning models with Keras, MachineLearningMastery.com. Available at: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/ (Accessed: 03 June 2023).

10. RBF SVM parameters. Available at: https://scikit-learn.org/stable/auto_examples/svm