Supervised Learning, Classification of Raisins Grown in Turkey

Ramkishore Rao/CS5033

Problem Outline: For our Machine Learning (IML) semester project component focusing on Supervised Learning (SL), I performed a classification of raisins grown in Turkey. The dataset that was used for this study contains seven (7) input features that are real variables that were used to predict whether the raisin grown is a Besni or Kecimen The data set was obtained from the UCI Repository[1]. We hypothesized that the algorithms that are implemented on this project will perform better than the accuracy that can be obtained by random classification of raisin type.

Dataset Features: The dataset for the raisins contains 7 input features, which are: Area, Perimeter, Major Axis Length, Minor Axis Length, Eccentricity, Convex Area, and Extent. These input features were derived from image processing of the grains. The dataset contains 900 examples.

Data Preparation: A summary of the dataset with its mean and standard deviation is presented in Table 1 below.

Table 1: Dataset Summary

Attribute	Mean	Std Deviation
Area	87804.13	39002.11
Major Axis Length	430.93	116.04
Minor Axis Length	254.49	49.99
Eccentricity	0.78	0.09
Convex Area	91186.09	40769.29
Extent	0.70	0.05
Perimeter	1165.91	273.76

Because of the wide disparity in the range of values for the attributes, the dataset was standardized per equation below.

$$\text{Standardized Value} = \frac{\text{Value} - \text{Mean}}{\text{Std Deviation}}$$

The list of standardized values was then discretized into test, validation, and train sets. The test set contained 20 pct of the values (180 examples) and the validation and train sets contained 108 and 612 examples, respectively. The target labels were relabeled as 0 and 1 for "Besni" and "Kecimen", respectively.

Algorithms Utilized: Several classification algorithms were evaluated to identify their ability to classify the grains. These algorithms were Stochastic Gradient Descent for Logistic Regression, K Nearest Neighbors, Naïve Bayes, Artificial Neural Networks, and Support Vector Machines (SVM). For all these algorithms, except SVM, code was developed for this project. sklearn package was used for SVM. A brief outline of the algorithms utilized for the classification is presented below:

Stochastic Gradient Descent: This approach applies initial random values to the coefficients (w_i) of the dependent variable (xi) and updates the coefficients based on the learning rate (l_rate) and the error in the prediction between the true label (yi) and the predicted label for each example. Sigmoid filter is applied to the linear combination of inputs to compute the predicted label (hw(xi)) for each example. l_rate of 0. 1 and 1000 iterations were utilized in this study.

Weights are updated according to the following rule:

$$w_i = w_i - \text{l\_rate} * (y_i - hw(xi)) * hw(x_i) * (1 - hw(x_i)) * x_i$$

K Nearest Neighbors: This algorithm classifies a train, validation, or a test datapoint by finding the labels of its K nearest neighbors. 4 nearest neighbors are used in this study. And the predicted label is the average of the true labels of those neighbors rounded up to 1 if prediction is >0.5 and rounded down to 0 if prediction is < 0.5.

Naïve Bayes: Naïve bayes classification follows the following algorithm:

$$y_{MAP} = \text{argmax}\ y \mathcal{E} Y\ [P(x|y)\ P(y)]$$

where x is the dependent variable; y is the class and y_{MAP} is the maximum a posteriori (MAP) hypothesis.

Artificial Neural Networks: An artificial neural network (ANN) with a multi-layer perceptron was used for this study. Two architectures were evaluated: 2 hidden layers, with 10 neurons each and one hidden layer with 20 neurons; both had an output layer with 2 neurons. The activation filter applied to the linear combination of inputs was the sigmoid filter. Back-propagation with stochastic gradient descent was utilized to update the weights of the coefficients for the network. Learn rate of 0.5 and 2000 iterations were utilized to train the network. RMSE errors were higher for the 2 hidden layer model, and accordingly predictions from the 1 hidden layer model are presented in this study.

Plots of RMSE errors during training for SGD and ANN are presented on Figures1 and 2 and 3, respectively.

SVM: sklearn's SVM-SVC package with linear kernel was utilized for this study. Its implementation is to provide a comparison to the results from the other algorithms. Dataset was not standardized during this implementation.

Evaluation of Algorithms: Confusion matrix was developed for each model for both the train and the test datasets. The format of the confusion matrix is as follows:

Table 2: Confusion Matrix Format

		Predicted	
		Besni	**Kecimen**
Actual	**Besni**	TN	FP
	Kecimen	FN	TP

TN = True Negative; FP = False Positive
FN = False Negative; TP = True Positive

[1] Source: https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset Set

Additional performance metrics included accuracy, precision, recall, and F-1 score. Higher values of these performance metrics indicate better model performance.

Accuracy is defined as the ratio of all the correctly classified data points to the total data points.

Accuracy = Number of correct predictions / Total number of predictions.

Precision is defined as the fraction of the correctly classified positive out of all the predicted positive points.

Recall is defined as the fraction of the correctly classified positive points out of all the actual positive points.

$$\text{Precision} = TP/TP+FP$$

$$\text{Recall} = TP/TP+FN$$

F1 score is needed when we want to strike a balance between precision and recall.

$$F1_{\text{Score}} = 2 \text{ x } \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Root Mean Square Error (RMSE) is a measured as follows and it is the standard deviation of the residuals:

$$\text{RMSE} = \text{sqrt}\left(\frac{1}{m}\sum_{1}^{m}\left((h(x_i) - y_i\right)^2\right)$$

Results: Evaluation of the various classification models relative to the performance metrics is presented in Tables 3, 4, and 5 below:

Confusion Matrix: Confusion matrix in the format presented in Table 2 for the various classification models on the test dataset are presented below. Of the 4 models for which code was developed, for the test dataset, the best TPR, accuracy, recall, F1_score, and lowest RMSE were obtained for the ANN model. Best precision was obtained for the KNN model for the test dataset. Also note that the estimates compare well with the SVM model developed with sklearn, except for the true negative rates for which the estimates were lower.

Table 6-1: SGD Confusion Matrix

	Besni	Kecimen
Besni	[[66.	19.]
Kecimen	[7.	88.]]

Table 6-2: Naïve Bayes Confusion Matrix

	Besni	Kecimen
Besni	[[61.	24.]
Kecimen	[6.	89.]]

Table 6-3: KNN Confusion Matrix

	Besni	Kecimen
Besni	[[70.	15.]
Kecimen	[18.	77.]]

Table 6-4: ANN Confusion Matrix

	Besni	Kecimen
Besni	[[64.	21.]
Kecimen	[3.	92.]]

Table 6-5: SVM Confusion Matrix[2]

	Besni	Kecimen
Besni	[[84.	11.]
Kecimen	[11.	74.]]

Area Under Curve (AUC): Receiver Operating Curves of true positive rates vs false positive rates were plotted for 1001 probability thresholds between 0 and 1 for the 1 hidden layer ANN architecture (Figure 4), for SGD (Figure 5), and for the SVM (Figure 6) for the test dataset. AUC results for these plots were similar for the models, with the AUC values ranging from 0.92 to 0.93.

Comparison with Related Work: Model results compare favorably with results reported on the UCI repository for this dataset by the original researchers[3]. For the 4 models for which code was developed, ANN generated better predictions on the test dataset, which is consistent with other studies[4]. Note, however that during training, the ANN appears to require a higher learning rate than SGD and more iterations to achieve similar RMSE scores. It is conceivable that the initial random values for weights may need to be tuned for better results. The impact of the starting values for the weights on model tuning is identified by other researchers[5].

Future Work: The developed models perform well when compared with the SVM model developed using the sklearn package. In order to further understand how to tune the ANN model, further analysis should be performed in future studies for the initial weights assigned to the links in the neural network and the effect of varying learning rates on back-propagation errors. The developed code should also be applied to datasets that contain other types of input variables, including categorical variables. It should also be evaluated on multi-class classification problems to test its performance.

Conclusion: In this study, various classification algorithms, SGD, KNN, Naïve Bayes, ANN, and SVM were applied to classify the grains. Accuracy of predictions were higher than 80 pct for all the models, with the highest score of 87.8 pct obtained from the SVM model for the test dataset. The models for which code was developed also compared favorably to the SVM prediction, with ANN with 1 hidden layer of 20 neurons resulting in an accuracy of 86.7 pct and SGD, logistic regression resulting in an accuracy of 85.5 pct, both for the test dataset. AUC for the ROC curve was more than 0.9 for ANN, SGD, and SVM, well above the AUC for random classifier of 0.5. The RMSE at the end of the training of both ANN and SGD models (~0.45) were only marginally higher than the RMSE's following conversion to binary values (0.365 for ANN and 0.380 for SGD), which demonstrates the merit of the developed models.

[2] Different seed and breakout of datasets per sklearn.

[3] Classification of Raisin Grains Using Machine Vision and Artificial Intelligence Methods, Cinar et. al, 2020

[4] The Hintons in your neural network, A quantum field theory of deep learning, Bondesan, 2021

[5] The elements of statistical learning, Hastie, et al. May 2001

Table 3: Performance of Models on Train Dataset

Perfor- -mance Metric	Logist -ic Regre -ssion (SGD)	Naïve Bayes	KNN	ANN[1]	SVM
Accuracy	0.879	0.843	0.897	0.863	0.856
Precision	0.860	0.795	0.913	0.813	0.856
Recall	0.900	0.917	0.873	0.934	0.850
F_1 Score	0.880	0.852	0.893	0.871	0.853
True Positi- -ve Rate	0.900	0.917	0.873	0.934	0.850
True Nega- -tive Rate	0.858	0.771	0.919	0.791	0.862
RMSE	0.348	0.396	0.321	0.370	0.379

Table 4: Performance of Models on Validation Dataset

Perfor- -mance Metric	Logist -ic Regre -ssion (SGD)	Naïve Bayes	KNN	ANN[1]	SVM
Accuracy	0.907	0.880	0.861	0.907	0.907
Precision	0.892	0.836	0.867	0.867	0.935
Recall	0.926	0.944	0.852	0.963	0.906
F_1 Score	0.909	0.887	0.860	0.912	0.921
True Positi- -ve Rate	0.926	0.944	0.852	0.963	0.906
True Nega- -tive Rate	0.889	0.815	0.870	0.852	0.909
RMSE	0.304	0.347	0.372	0.304	0.304

Table 5: Performance of Models on Test Dataset

Perfor- -mance Metric	Logist -ic Regre -ssion (SGD)	Naïve Bayes	KNN	ANN[1]	SVM
Accuracy	0.855	0.833	0.816	0.867	0.878
Precision	0.822	0.788	0.837	0.814	0.871
Recall	0.926	0.937	0.811	0.968	0.871
F_1 Score	0.871	0.856	0.824	0.885	0.871
True Positi- -ve Rate	0.926	0.937	0.811	0.968	0.871
True Nega- -tive Rate	0.776	0.718	0.823	0.753	0.884
RMSE	0.380	0.408	0.428	0.365	0.350

Note 1: Results from ANN Architecture of 1 Hidden Layer and 20 Neurons Presented

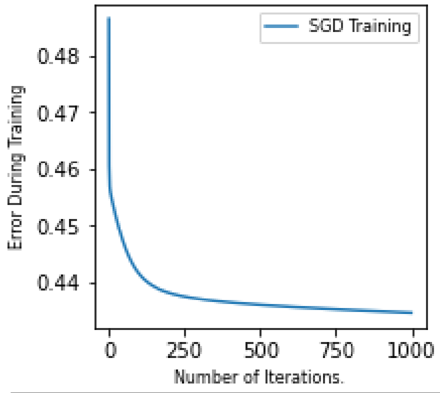

Fig 1: SGD Training Errors

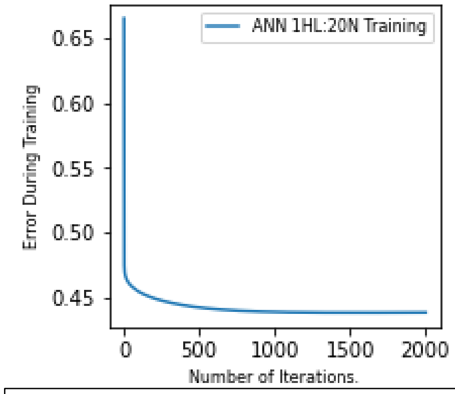

Fig 2: ANN 1 HL Training Errors

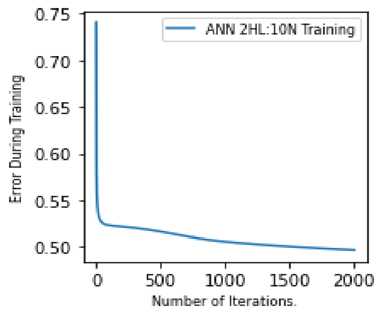

Fig 3: ANN 2 HL Training Errors

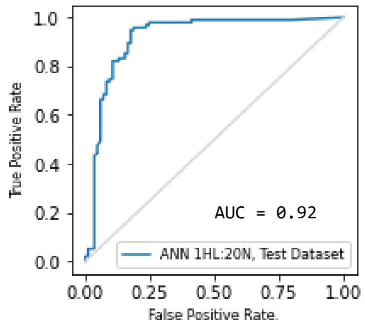

Fig 4: ANN, Test, 1 HL ROC Curve

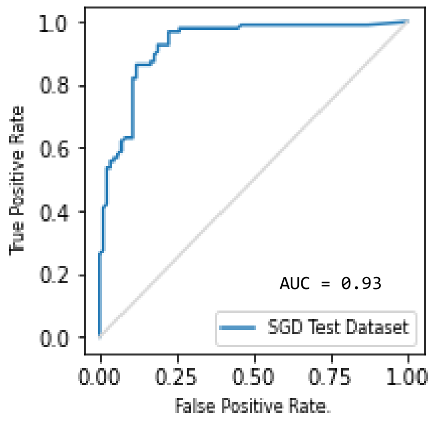

Fig 5: SGD, Test, ROC Curve

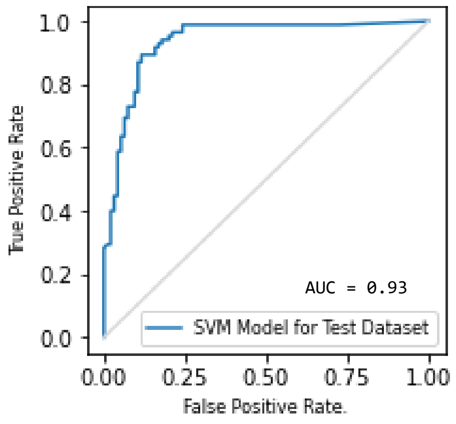

Fig 6: SVM, Test, ROC Curve