Comparison of Supervised Learning Algorithms

Andreas Koutras

I. Introduction

This project investigates the behavior of k-nearest neighbors (KNN), support vector machines (SVM), and neural networks (NN) in classification problems. The algorithms are evaluated using two datasets: the “Vehicle Silhouettes” dataset and the “Default of Credit Card Clients” dataset, both of which are accessible from the UCI Machine Learning Repository. For brevity, we will refer to these as the Vehicle dataset and the Default dataset, respectively.

These two datasets are thought to be interesting for this study due to their different sample sizes, feature types, and class proportions. The Vehicle dataset consists of 845 samples with 18 non-discrete integer valued features and a target vector containing four evenly balanced classes each representing a different type of vehicle. The features are geometric measurements derived from images, including elongation, aspect ratios, skewness, etc. In contrast, the Default dataset consists of 29,965 samples with 23 mixed-type features. Among these, one feature is binary (sex), two are categorical (education level and marital status), six are discrete integers representing payment delays in moths (ranging from 1 to 9), and the remaining features are integers representing dollar amounts for credit limits, bill statements, and past payments. Additionally, the class distribution differs from the Vehicle dataset. The Default dataset has a binary class variable, with values of 1 and 0 indicating whether a client has defaulted or not of a client unlike the previous dataset. The class proportions are imbalanced, with class 0 representing 78% of the samples and class 1 only 22%.

Minimal preprocessing was performed on the data. Initially, we removed duplicate samples and those with NaN entries, resulting in the final sample sizes noted above. In the Default dataset, we applied one-hot encoding to the marital status feature, transforming it into three binary features (married, single, others) to prevent the algorithm from assuming a hierarchy among the values. In contrast, one-hot encoding was deemed unnecessary for the education level feature, as its numeric values (ranging from 1 to 4) already reflect a clear hierarchy, with 1 representing the highest education level. After one-hot encoding, the total number of features in the Default dataset increased to 25. Finally, we generated box plots for each feature, which indicated several outliers, particularly in the bill and payment amount features. Although removing these outliers could enhance model performance, we chose to retain them to evaluate the algorithms’ behavior under the challenges of the original datasets.

To determine the important features for predicting the class value, we used two independent methods: the correlation matrix and a decision tree employing the Gini index as the split criterion. Both methods revealed that the most significant feature in the Vehicle set is the Elongatedness feature, while in the Default set, the most significant is the feature of the previous month’s payment status (feature X6). The absolute correlation coefficients with the target variable for these features are 0.33 and 0.32 respectively. Additionally, the correlation matrices for both datasets indicated the presence of several strongly intercorrelated features, with correlation coefficients reaching as high as 0.97.

Considering the characteristics of the datasets, we anticipate the following:

- KNN may struggle to perform effectively due to the high dimensionality of the datasets, which makes it sensitive to the curse of dimensionality.
- The presence of intercorrelated features could complicate the models and increase computational costs without adding significant discriminative power.
- The SVM algorithm is expected to perform well if a clear decision boundary can be modeled with an appropriate kernel. However, it is also expected to be computationally expensive when applied to the Default dataset due to its large sample size.
- The NN may outperform the other two algorithms due to its numerous hyperparameters that can be fine-tuned for optimal results.

II. Experimental Methodology

The NN, KNN, and SVM algorithms are sensitive to features scaling. In these algorithms, features with larger numerical values can dominate the solution compared to those with smaller numerical values. To address this, we standardized all feature columns in both datasets so that each standardized feature has zero mean and unit variance. This was performed using the mean and variance from the training set, and then the same transformation was applied to the test set. We split the data, allocating 70% for training and 30% for testing, with a random stratified split to maintain the class proportions of the full dataset. The models were trained on the training set and tested on the test set. All tuning and experimentation were conducted exclusively on the training set, while the test set was reserved for reporting the final performance scores of the models. To ensure reproducibility of the results, we used a fixed random seed throughout the study.

The Scikit-learn (sklearn) library in Python was used to build the models and conduct the experiments. For tuning the hyperparameters and generating the learning curves, we used (stratified) K-fold cross-validation with five folds. This means that in each fold, 80% of the training data was used for training, while 20% was reserved for testing to determine the validation score. The final training and cross-validation scores (referred to as the "test score" for simplicity) were averaged across the five folds, with standard deviations calculated to capture the variance in performance. The performance score of the algorithms was evaluated using the area under the curve of the receiver operating characteristics curve (ROC-AUC), which has the advantage of considering different values of the probability decision threshold. The graphs of the following sections show the average ROC-AUC scores and the range of one standard deviation above and below the average.

III. Hyperparameter tuning

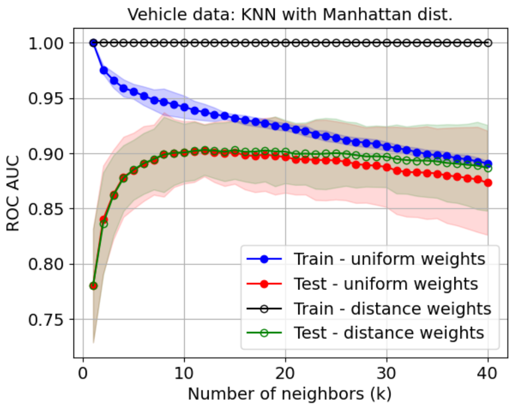
A. *KNN model*

The hyperparameters we examined for the KNN classifier are the number of neighbors (k), the distance metric, and the weighting scheme for the neighbors. For the distance metric we considered the Euclidean distance and the Manhattan distance. For the weighting scheme, we considered uniform weights and weights inversely proportional to the distance metric. Fig. 1 presents the cross-validation results for both datasets, displaying training and test scores as a function of k for the two weighting methods (uniform and distance-based) using the Manhattan distance. Since the curves for Euclidean distance were very similar, they are omitted for brevity. It is worth noting that the peak test scores with Euclidean distance were ever slightly lower than those achieved with Manhattan distance. From these results, a number of interesting observations can be drawn.

As shown in the figure, the use of distance-based weights in KNN leads to a perfect training score for any value of k. This is because the distance between training example and itself is zero, leading to an infinite weight. Consequently, the training example is always classified correctly, regardless of its neighbors.

For small values of k, particularly k=1, 2, or 3, we observe a combination of low test score and high training score, indicating high variance and low bias. This happens because the model overfits the training data, creating complex decision boundaries, that fail to generalize well to unseen data. As k increases approaching its optimal value, the model overfits well leading to better generalization and a reduced the variance (gap) between the training and test scores. However, when k exceeds the optimal value, the bias increases, resulting in lower scores for both the training and test sets, as seen in the Vehicle dataset. Interestingly, in Default dataset, the training score with test scores converge to the same score value even with k values up to 8000 when using uniform weights. This behavior could be due to several factors: 1) class imbalance, making it more likely to have more neighbors from the dominant class, 2) lack of clear boundaries or clustering between the two classes making it difficult to separate the two classes effectively, 3) sparse data points or irrelevant features that reduce the discriminative power of the distance metric.

It also worth noting that, in Vehicle dataset's validation curve, when k exceeds the optimal value, the test score decreases more gradually with distance-based weights compared to uniform weights. This occurs because, with distance-based weights, the closer neighbors contribute more to the prediction, yielding results similar to those of uniform weights but with a smaller k.



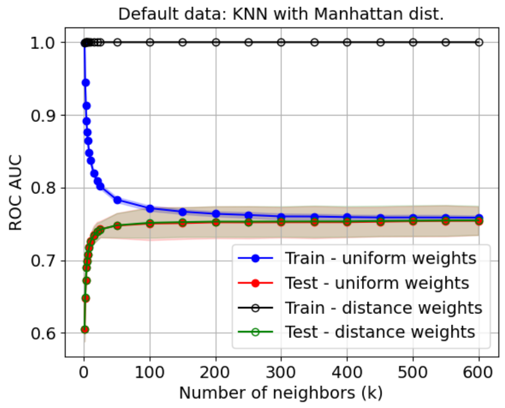


Fig. 1: KNN model cross-validation curves varying the k number of neighbors for the Vehicle (top) and Default (bottom) datasets comparing the training and test ROC-AUC scores for uniform weights and distance-based weights.

In the final KNN model training (presented in a later section), we selected the Manhattan distance with distance-based weights, as this combination achieved slightly higher test scores across both datasets. For the Vehicle dataset, we chose k=12, corresponding to the peak ROC-AUC test score of 0.903 and a test accuracy for of 0.695. For the Default dataset, we opted for k=200, where the test score has nearly reached its maximum, with a ROC-AUC of 0.753 and accuracy of 0.804. It is important to note that larger values of k increase the computational demand of the querying process.

B. *The SVM model*

The hyperparameters we considered for the SVM classifier are the kernel function and the regularization parameter C. Choosing an appropriate kernel function is crucial for enabling the SVM to capture non-linear decision boundaries, since the default linear kernel may be insufficient for such problems. Additionally, fine tuning the regularization parameter C is important for controlling the bias-variance trade-off in the

SVM. In the implementation of sklearn, the regularization strength is inversely proportional to C. A high value of C places greater emphasis on classifying correctly all the training examples, which increases the risk of overfitting (i.e., a model with low bias and high variance). Conversely, a lower C allows for some misclassifications in favor of a simpler decision boundary that is more likely to generalize well to unseen data (i.e., a model with higher bias and lower variance).

Fig. 2 shows the validation curves for tuning the kernel function in the Vehicle and Default datasets. The kernel alternatives we examined are the linear kernel, polynomial kernels (poly) of 2nd to 8th degree, the radial basis function (RBF) kernel, and the sigmoid kernel. For the kernel coefficient parameter gamma of the poly, RBF, and sigmoid kernels in sklearn and for the bias term r we used the default settings of sklearn. The regularization parameter C was taken equal to 1.0, which is the default value in sklearn. Based on the results of Fig. 2 we can comment the following. In the Vehicle dataset, the

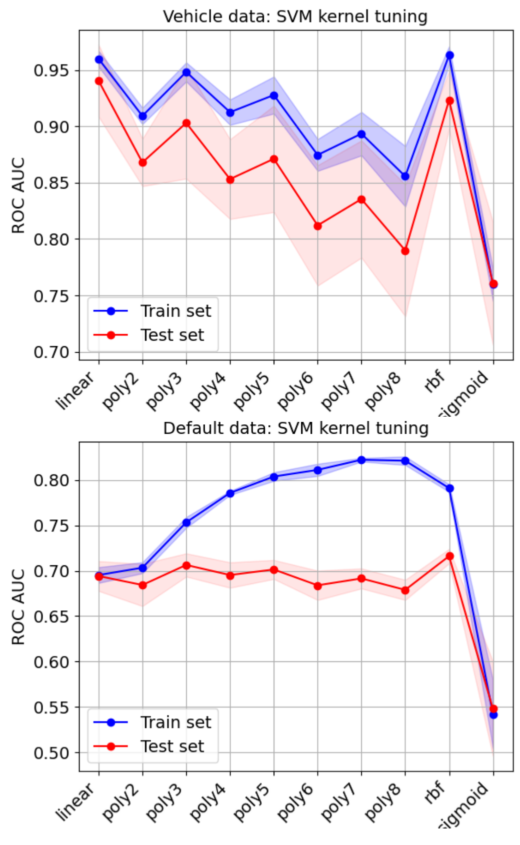


Fig. 2: SVM model cross-validation curves varying the kernel function for the Vehicle (top) and Default (bottom) datasets comparing the training and test ROC-AUC scores.

linear kernel achieves a higher training score, therefore lower bias, than the polynomial kernels. In fact, the training score decreases as the degree of the polynomial kernel increases, suggesting that the four classes are nearly linearly separable. The RBF kernel produces a similar training score to the linear kernel, demonstrating that it also captures the decision boundaries well, though with an added smoothing effect on the predictions. The test scores follow a similar trend to the training scores but with some variance in-between. Lastly, the sigmoid kernel has the worst performance and exhibits such a high bias that results in consistently low scores for both the training and test sets, with virtually no variance.

In the Default dataset, the model displays a different trend, reflecting the more complex nature of the class boundaries. The training score improves as the degree of the polynomial kernel increases, but this comes at the cost of higher variance for higher-degree polynomials. The RBF kernel achieves the best balance in the bias-variance trade-off, attaining a slightly lower training score than the 8th degree polynomial (poly8), but a higher test score.

Fig. 3 shows the validation curves for varying the regularization parameter C. For the Vehicle data, the results are presented for the linear and RBF kernels, which achieved the highest test scores in Fig. 2. The curves indicate that when the regularization is weak (i.e., with larger values of C, the model performs well with either kernel. However, as the regularization strength increases (smaller C values), the model's performance declines significantly. This supports our finding that the decision boundaries for this dataset are effectively captured by the linear and RBF kernels directly, without the need for regularization. The RBF kernel actually achieved a slightly higher test score than the linear kernel leading us to select the RBF kernel for the final model training. The RBF kernel attained its peak test score at C=10, with a ROC-AUC of 0.952 and an accuracy of 0.807.

For the Default dataset, Fig. 3 presents validation curves for the 3rd-degree polynomial and RBF kernels, which achieved the highest test scores in Fig. 2, along with the 8th-degree polynomial, which exhibited the lowest bias but highest variance. Unlike the Vehicle dataset, we observe that for large values of C (and higher-degree polynomials), the model tends to overfit the training data. Conversely, when C is too low, the model becomes overly biased, as the regularization term dominates, diminishing the influence of the kernel function. The test scores follow a nearly inverted U-shaped trend, indicating an optimal C value that balances the bias-variance trade-off. The RBF kernel achieved the peak test score at C=0.0001, which was selected for the final model training. The corresponding cross-validation ROC-AUC was 0.72, with an accuracy of 0.779. The need for stronger regularization is another evidence of the complex decision boundary in the Default dataset.

C. The Neural Network model

The multi-layer perceptron classifier (MLP) of sklearn was used. The hyperparameters examined are the number of hidden layers, the number of nodes per hidden layer, the type of the activation function (logistic, ReLU, tanh), and the regularization factor alpha. We limited our analysis to neural networks with either one or two hidden layers, and for the models with two hidden layers, we assumed the same number of nodes in each layer. In the MLP implementation of sklearn the strength of regularization is proportional to the value of

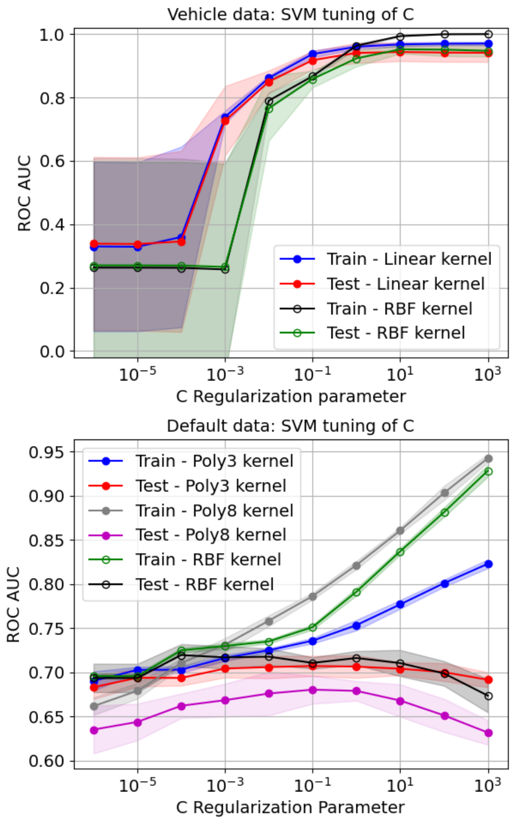


Fig. 3: SVM model cross-validation curves varying the C regularization parameter for the Vehicle (top) and Default (bottom) datasets comparing the training and test ROC-AUC scores for different kernel functions.

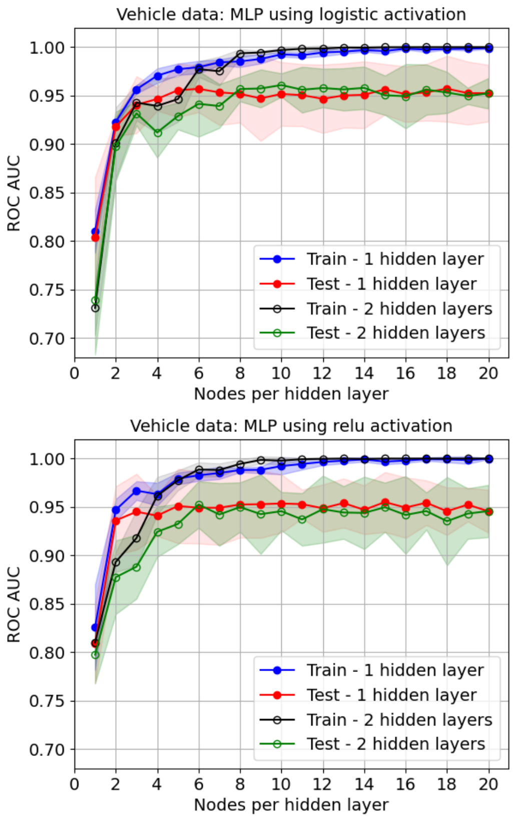


Fig. 4: MLP model cross-validation curves for the Vehicle dataset varying the number of nodes per hidden layer using logistic activation (top) and ReLU activation (bottom). The training and test ROC-AUC scores are compared for models with a single and models with two hidden layers.

alpha.

1) Model tuning for the Vehicle dataset

Fig. 4 presents the cross-validation curves for the Vehicle dataset, comparing the performance of MLP models with one and two hidden layers using two activation functions: logistic and rectified linear unit (ReLU). The curves depict the relationship between the number of nodes per hidden layer and model performance. The regularization factor alpha was set to 0.0001, which induces a small regularization effect and is the default value in sklearn. To optimize the neural networks, the stochastic gradient-based optimizer ADAM was employed with a convergence tolerance of 1e-6, a maximum of 4000 iterations, and the default initial learning rate of 0.001 in sklearn. These parameters were found to produce convergent solutions. The plots for the tanh activation function were excluded to conserve space, as their inclusion would not alter the overall conclusions.

As shown in Fig. 4, the two-hidden-layer model performs worse than the single-hidden-layer model when the number of nodes is small (fewer than 4 or 6 nodes per layer, depending on the activation function). The second hidden layer introduces unnecessary additional nonlinearity increasing the model bias for the Vehicle dataset. However, as the number of nodes increases, the performance of the two-hidden-layer model improves, as the added nodes provide more degrees of freedom, helping to reduce the bias. The model's nonlinearity also depends on the activation function used. The results clearly show that the single-hidden-layer model delivers better performance, achieving higher test scores with fewer nodes. This aligns with our observations from the SVM model for this dataset, where the linear kernel outperformed the polynomial kernel.

Based on the results, the peak test score was achieved by the MLP with a single hidden layer, 6 nodes, and the logistic activation function. We therefore adopted this configuration for the final model training, yielding cross-validation scores of ROC-AUC = 0.957 and accuracy = 0.816. While the cross-validation curve with respect to the regularization factor alpha was also obtained, it is omitted here for brevity. The curve exhibits a similar trend to the regularization parameter in SVM (with the horizontal axis direction inverted), where model performance deteriorates as regularization increases. The highest test scores are achieved and maintained for alpha¡0.01, confirming our choice of alpha=0.0001.

2) Model tuning for the Default dataset

Fig. 5 shows the cross-validation curves for the Default dataset, illustrating the performance of MLP models with one and two hidden layers as the number of nodes per layer increases. The ReLU activation function was used, and alpha was set to 0.0001. Unlike in the Vehicle dataset, the two-hidden-layer model fits the training data better, or at least as well as the single-hidden-layer model for small number of nodes. However, as the number of nodes grows, the two-hidden-layer model tends to overfit the training data sooner and cannot generalize. Since both models performed similarly on the test data when the number of nodes was small (fewer than 7 nodes per layer), we conducted a grid search over the number of hidden layers, nodes per layer, and values of the regularization factor alpha to determine the best configuration for test performance. The grid search focused on models with 3 to 6 nodes per layer, also considering architectures with fewer nodes in the second hidden layer than the first. The results showed that while the two-hidden-layer model with 6 nodes per layer achieved the highest test score, the improvement was marginal compared to the simpler single-hidden-layer model with 3 nodes. The cross-validation ROC-AUC scores were 0.772 and 0.770, respectively. To prioritize simplicity, we selected the single-hidden-layer MLP with 3 hidden nodes and an optimal alpha value of 0.01. Additionally, to assess the effect of different activation functions on the selected MLP model, we compared the training and test using the activation functions: identity, logistic, tanh, and relu. The training ROC-AUC scores were 0.723, 0.781, 0.780, and 0.778, respectively, while the test scores were 0.721, 0.767, 0.766, and 0.770, respectively. The highest training score was achieved with ReLU function matching our initial guess.

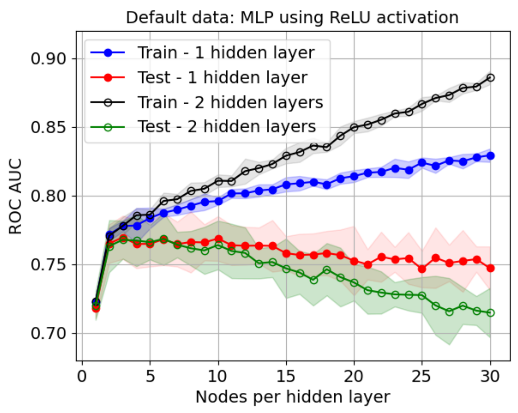


Fig. 5: MLP model cross-validation curves for the Default dataset varying the number of nodes per hidden layer using the ReLU activation function. The training and test ROC-AUC scores are compared for models with a single and models with two hidden layers.

3) Model convergence

An essential aspect to evaluate is the convergence behavior of the selected models for both datasets with respect to the number of iterations of the MLP solver. Fig. 6 illustrates the evolution of ROC-AUC scores as the solver's maximum number of iterations increases. To ensure that the solver runs for the full set of iterations, the convergence tolerance was set to a very small value to avoid early termination. As depicted in the plots, the scores eventually converge to their final values for both datasets. The required number of iterations to reach convergence differs significantly: 4000 for the Vehicle dataset and 200 for the Default dataset. This difference arises because the ADAM solver was used for the Vehicle dataset, while the LBFGS solver was employed for the much larger Default dataset, as ADAM proved considerably slower for that case.

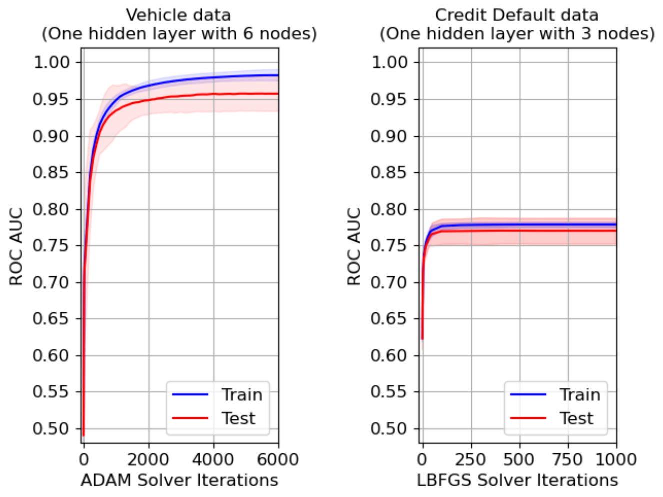


Fig. 6: Iterative learning curves for the MLP models of the Vehicle (left) and Default (right) datasets.

IV. Comparison across models and datasets

A. Learning Curves

In this section, we examine the scalability of the algorithms with respect to the size of the training set using the hyper-parameters and solver settings tuned in the previous section. Fig. 7 and Fig. 8 show the learning curves of the three algorithms for the Vehicle and Default datasets respectively. The algorithms were trained for different training set sizes but were tested on the same validation set. A stratified 5-fold cross-validation was used.

For the Vehicle dataset, the MLP model demonstrates superior performance, achieving a slightly higher test score and lower variance compared to the SVM, which holds the second-highest score. Unlike the SVM and KNN models, whose test curves have visibly converged to a final value, the MLP model shows potential for further improvement with a larger training set, as indicated by the sustained upward trend in the test score. The sudden dip in the MLP's score at 400 training samples is likely due to problematic convergence of the ADAM solver. In the SVM model, the lower training score observed with fewer than 100 training samples is likely a numerical artifact of the approximate method used in sklearn's SVM to estimate class probabilities. It appears that this method struggles with small sample sizes. Since ROC-AUC relies on class probability estimates, this issue impacts the score. In contrast, the corresponding training accuracy score, which does not depend on probability estimates, exhibits the expected monotonically decreasing trend. In the Default dataset, all

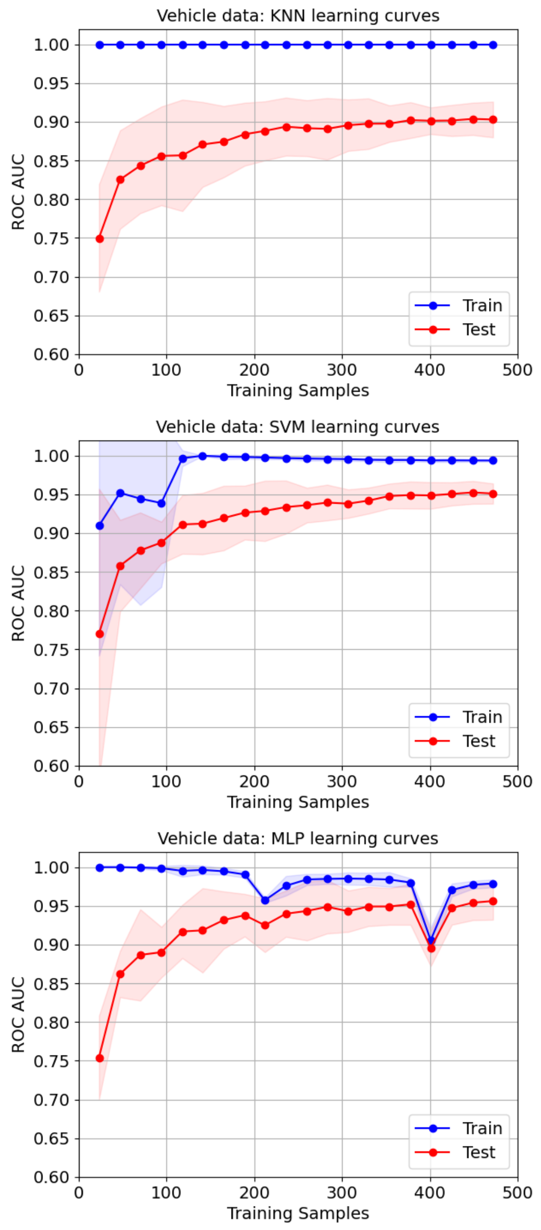


Fig. 7: Learning curves for the models of the Vehicle dataset: KNN (top), SVM (middle), MLP (bottom). The number of samples used for training is varied while the samples for testing are kept the same.

three models display high bias, achieving considerably lower scores compared to their performance on the Vehicle dataset. The results suggest that neither of the three models would significantly benefit from a larger training set. In fact, the SVM model experiences a slight decline in test score when the training size exceeds 15,000, likely due to overfitting to noise. Nonetheless, both the MLP and SVM models exhibit very low variance as the number of training samples increases, indicating stable generalization.

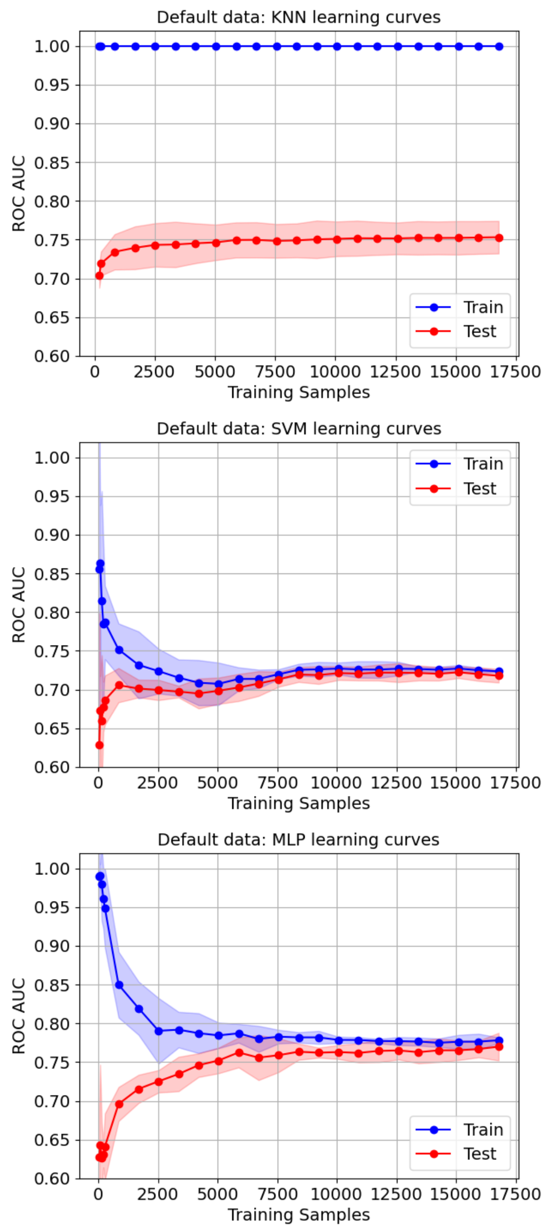


Fig. 8: Learning curves for the models of the Default dataset: KNN (top), SVM (middle), MLP (bottom). The number of samples used for training is varied while the samples for testing are kept the same.

B. *Final Training and Testing*

After tuning the models, we trained them on the full training set and evaluated their performance using the original test set, which comprised 30% of the sample data and was reserved for this purpose. Tables I and II provide a comparison of the

ROC-AUC scores for both the training and test sets in the Vehicle and Default datasets respectively, as well as the time elapsed during training and querying the training set for each model. The performance scores align with our observations from previous sections. In terms of timing, we can draw several conclusions. KNN, being an instance-based learner, exhibits significantly lower training time compared to querying time, as its training phase merely involves storing the dataset in memory. Querying time increases with both the number of neighbors and the dataset size, as reflected in the longer query time for the Default dataset. Conversely, the MLP model incurs a higher computational cost during training due to weight optimization, but its querying phase is nearly instantaneous, as it relies on precomputed weights. For the SVM model, training becomes notably expensive in the larger Default dataset, but querying is even more computationally costly. This is because, during training, the SVM only uses a subset of data points, the support vectors, to optimize the decision boundary. However, during querying, each data point must be com-pared with all the support vectors, which can be time-consuming, especially when the number of support vectors is large, as seen in the Default dataset.

TABLE I: Final training and testing for the Vehicle dataset models

Model	Hyper-parameters	Training set roc-auc	Test set roc-auc	Train time (s)	Query time (s)
KNN	k=12, Manhatt. dist., weights=dist.	1.000	0.909	0.005	0.074
SVM	kernel=rbf, C=10.0	0.993	0.949	0.016	0.022
MLP	hid_layers=1, nodes=6, activ.=logist., alpha=1e-4, solver=ADAM	0.979	0.951	3.477	0.001

TABLE II: Final training and testing for the Default dataset models

Model	Hyper-parameters	Training set roc-auc	Test set roc-auc	Train time (s)	Query time (s)
KNN	k=200, Manhatt. dist., weights=dist.	1.000	0.761	0.005	2.237
SVM	kernel=rbf, C=1e-4	0.721	0.723	9.919	801.7
MLP	hid_layers=1, nodes=3, activ.=ReLU, alpha=0.01, solver=LBFGS	0.777	0.780	3.254	0.002

V. Conclusions

In this study, we evaluated the performance of KNN, SVM, and MLP models on the Vehicle and Default datasets. For the Vehicle dataset, all three models delivered satisfactory results, despite initial concerns that the limited training sample size (590 samples) might be insufficient given the 18 features involved. The strong performance, particularly by the SVM with a linear kernel, indicated that the data was likely linearly separable. Among the models, MLP and SVM outperformed KNN in this dataset. In contrast, for the Default dataset, all models struggled, underfitting even the training data. The MLP still had the best performance, but the SVM performed worse than the KNN model and its high computational cost make SVM impractical for this dataset. Furthermore, despite the presence of several intercorrelated features, the MLP solver was able to extract simpler models that matched or exceeded the performance of more complex ones.

The challenges with the Default dataset include imbalanced classes (1:3.5 class ratio), intercorrelated features, and the presence of outliers. The curse of dimensionality was not a significant factor, as the learning curves indicated that increasing the sample size would not result in improvement. To enhance future results, dimensionality reduction techniques, such as Principal Component Analysis or feature selection using decision trees, along with outlier removal, could be explored. Additionally, ensemble methods like boosted decision trees may help reduce misclassification errors.