Machine Learning
Supervised Learning

Jorge Salvador Aguilar Moreno

***Abstract*—This assignment analyzes two datasets using three supervised learning techniques: (1) neural networks, (2) support vector machines, and (3) k-nearest neighbors. The objective is to implement and compare the performance of these algorithms, examining their results on two datasets: one related to diabetes diagnosis and the other concerning concrete compressive strength prediction.**

***Index Terms*—supervised learning, machine learning, neural networks, knn, nearest neighbor, SVM, support vector machine, cross-validation.**

I. Introduction

The objective of this Georgia Tech course is to develop a deep understanding of machine learning (ML), a discipline that studies and develops algorithms that learn from data [1]. Machine learning can be broadly divided in three main categories: supervised learning, unsupervised learning, and reinforcement learning. This report will focus on supervised learning, which forms the basis of this first assignment.

Supervised learning entails training models with known features and corresponding labels. In other words, information regarding the input and the associated outputs is given. Different algorithms can be used using supervised learning techniques with the goal of predicting the correct output given the inputs (also called features). For this assignment, we will explore three popular supervised learning algorithms: neural networks, support vector machines, and k-nearest neighbors.

These algorithms are particularly effective in solving classification problems, where the objective is to predict a class or category based on the input features [2].

This report is organized as follows: first, the datasets are introduced, along with a discussion of their relevance and potential to yield insightful results. Next, initial hypotheses are outlined, setting the expectations prior to implementing the supervised learning algorithms. The experimental methodology is then presented, detailing the general approach applied consistently across the three algorithms. Following this, the results of each learning algorithm are presented, supported by graphs that illustrate their performance. Finally, the algorithms are compared, and key findings are summarized in the conclusion.

II. Datasets

The following datasets were used as the basis for implementing the supervised learning algorithms in this project:

1) **Concrete compressive strength dataset** (UCI Machine Learning Repository) [4]. This dataset contains 1,030 instances and 8 features: cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, and age.
2) **Diabetes dataset** (Kaggle) [3]. This dataset contains 768 instances and 8 features: pregnancies, glucose level, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age.

These datasets are interesting for this project for several reasons:

- **Different classification problems**: The diabetes dataset is a binary classification problem, where the goal is to predict the presence or absence of diabetes. On the other hand, the concrete compressive strength dataset, originally a regression problem, was transformed into a multi-class classification task by binning the continuous output into strength ranges. This allows us to compare algorithm performance on both binary and multi-class classification problems.
- **High output variability**: Both datasets present complex problems where the outputs are challenging to predict with high accuracy. For example, medical diagnosis and concrete strength estimation likely depend on additional features not captured in these datasets. Nonetheless, our objective is to implement algorithms that can achieve reasonable predictive performance despite these complexities.
- **Real-world data**: These datasets originate from real-world sources, introducing practical challenges such as noise and errors in the data. This reflects realistic conditions for applying machine learning algorithms, where perfect accuracy is often unattainable due to the inherent uncertainty in unseen data.

The datasets required minimal pre-processing, as they did not contain significant outliers or missing values. All instances were deemed useful, and no major data cleanup was necessary. This made the datasets highly usable for our experiments.

It is important to note that these datasets are not part of the prohibited list mentioned in the assignment instructions. While there is a diabetes dataset available in the `sklearn.datasets` library, it contains entirely different information from the one used in this project.

III. Hypotheses

Based on the datasets selected for this study, the following hypotheses are proposed:

1) **Diabetes dataset prediction accuracy**: It is hypothesized that the accuracy of the predictions for the diabetes

dataset will exceed 85%. This expectation stems from the fact that binary classification problems, such as this one, inherently offer a 50% baseline chance of correct classification.

2) **Concrete compressive strength prediction accuracy**: For the concrete compressive strength dataset, it is hypothesized that the prediction accuracy will fall in the range of 30% to 40%. Given that this dataset has been transformed into a multi-class classification problem, and considering the complexity of predicting compressive strength, the accuracy is expected to be lower than that of the diabetes dataset.
3) **Algorithm performance on the diabetes dataset**: It is believed that the k-nearest neighbor (k-NN) algorithm will yield the highest accuracy for the diabetes dataset. This is based on the assumption that the medical diagnosis data could be naturally grouped into clusters in a multidimensional space.
4) **Algorithm performance on the concrete compressive strength dataset**: For the concrete compressive strength dataset, it is hypothesized that neural networks will provide the most accurate predictions. The complexity of the dataset and the non-linear relationships between features suggest that neural networks are likely to outperform other algorithms.

These hypotheses will be rigorously tested through experimentation, and the results will be supported with evidence in the following sections.

IV. Experimental Methodology

In this assignment, the three supervised learning algorithms are evaluated using a consistent experimental methodology. This systematic approach ensures a fair comparison between the models and provides evidence to either support or refute the initial hypotheses.

The experimental methodology consists of the following steps:

1) **Data Preprocessing**: The first step involves cleaning the dataset to handle any missing values or outliers, if present. This was not necessary in the two selected datasets. Categorical labels are encoded into numerical values where applicable. In the case of the concrete compressive strength dataset, the continuous target values are grouped into bins using predefined range limits: `bins = [0, 15, 25, 35, 45, 55, 65, 75, 85, 95, 105]`.
2) **Train-Test Split**: The dataset is divided into training and test sets. The training set is used for model training and hyperparameter tuning, while the test set is reserved for final evaluation to represent unseen data, ensuring that model performance is not biased by prior exposure to the test data.
3) **Feature Scaling**: Feature scaling is applied to ensure that all features contribute equally to the model, as it prevents certain features with larger ranges from dominating others. The same scaling is consistently applied to both the training and test sets to avoid data leakage and ensure valid evaluation.
4) **Algorithm Initialization**: Each supervised learning algorithm is initialized with hyperparameters that are expected to yield optimal results. These hyperparameters are tuned manually in the subsequent steps.
5) **Cross-Validation**: Cross-validation is performed on the training data to evaluate the model's performance. A 5-fold cross-validation technique is used, where the training data is split into 5 subsets, and each subset takes turns being the validation set. Accuracy is the primary evaluation metric, but other performance metrics, such as F1-score and the confusion matrix, are also considered to gain a more comprehensive understanding of the model's performance.
6) **Model Iteration**: Based on the cross-validation results, the model is iteratively refined. If the model's performance is satisfactory and accuracy is maximized, the process moves to the next step. If performance is suboptimal, hyperparameters are adjusted, and the model is re-evaluated, returning to step 4.
7) **Final Model Training**: Once the optimal hyperparameters are determined to manually maximize accuracy, the model is re-trained on the entire training dataset. The model is then evaluated using the same performance metrics as in the cross-validation stage. The expectation is that the final metrics will align closely with the cross-validation results, confirming the model's ability to generalize to unseen data.
8) **Plot Generation**: Various plots are generated to visualize the model's performance and compare different iterations. For neural networks, iterative learning curves are produced, plotting the number of iterations against log-loss and accuracy. For all supervised learning algorithms, learning curves (training size vs. accuracy) and validation curves (algorithm hyperparameters vs. accuracy) are generated to analyze performance trends.
9) **Grid Search**: Grid search is employed to explore a predefined grid of hyperparameters and identify the optimal parameter set through exhaustive search. This serves as a final step to compare manually tuned hyperparameters with those found using the automated grid search. Grid search was not used initially, as one of the key learning objectives of this assignment is to develop a deeper understanding of how hyperparameters impact model performance through manual tuning, rather than relying on automated methods to find the optimal solution.

V. Results from Algorithms

This section presents the results obtained from applying the experimental methodology to the three supervised learning algorithms. All algorithms are implemented using the `scikit-learn` library in Python. The primary performance metric used to evaluate the models is **accuracy**. Accuracy was selected for its simplicity and ease of interpretation, making it well-suited for balanced datasets, as presumed in this case.

Furthermore, accuracy can be uniformly applied across all models, allowing for a clear and direct comparison of their performance.

VI. Neural Networks (NN)

Neural networks are iterative algorithms that consist of layers of interconnected nodes (neurons) and learn by adjusting weights through multiple iterations to capture complex patterns in data [1].

Neural network algorithms are tested by varying key hyperparameters, including the number of nodes, the number of layers, activation functions, solver type, and learning rates. The results for each dataset are presented in the following subsections.

A. Neural Networks: Diabetes Dataset

The neural network implementation that achieved the highest accuracy through manual tuning is as follows:

```
MLP = MLPClassifier(
    activation='logistic',
    hidden_layer_sizes=(8, 8),
    solver='adam',
    alpha=0.0001,
    max_iter=600,
    random_state=sklearn_random_state)
```

This configuration uses the logistic activation function, a two-layer architecture with 8 nodes per layer, and the Adam optimizer. The regularization parameter α is set to 0.0001, and the model runs for a maximum of 600 iterations to ensure convergence.

The **iterative learning curves** in Figure 1 illustrate the variation of log-loss and accuracy as the number of iterations increases. The iteration vs. log-loss curve on the left shows a sharp downward trend initially, flattening at around 200 iterations, where the log-loss on the validation set stabilizes at approximately 0.48. This is where our scikit-learn model stops iterating, as further changes in loss are not greater than 0.0001 between iterations. After this point, the plot illustrates how the log-loss continues to decrease for the training set while increasing for the validation set, indicating that the model begins to overfit.

The iteration vs. accuracy curve in Figure 1 on the right demonstrates a rapid increase in accuracy during the initial iterations for both the training and validation sets, indicating that the model is learning quickly and achieving over 65% accuracy in early stages. From around 20 to 100 iterations, accuracy remains relatively stable, showing no significant improvements until a second, more gradual increase occurs. Finally, the curve flattens, suggesting the model has converged and additional iterations do not further enhance performance. This convergence occurs at an accuracy of around 78%.

In Figure 2, the **learning curve** on the left evaluates the model's performance based on the size of the training data, while the **validation curve** on the right assesses the effect of a

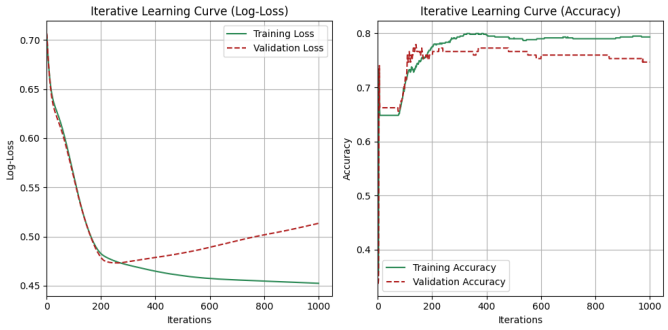

Fig. 1. Neural networks trained with diabetes dataset.
Iterative learning curves: log-loss (left) and accuracy (right)

single hyperparameter—specifically, the regularization parameter alpha. The training size vs. accuracy plot shows that the training and validation accuracies converge to approximately 78% when the training size reaches around 250 samples. Beyond this point, increasing the training size does not lead to significant changes in accuracy for either set, indicating that adding more data is unlikely to further improve the neural network's performance.

The validation curve demonstrates that for alpha values between 10^{-5} and 10^{-2}, the accuracy remains stable at approximately 78% for both the training and validation sets. However, when alpha exceeds 10^{-2}, the accuracy for both sets drops significantly, showing that the model is relatively insensitive to small values of alpha, but overly large values lead to a decline in performance. This suggests that regularization has minimal impact on the model's accuracy as long as alpha is kept below 10^{-2}.

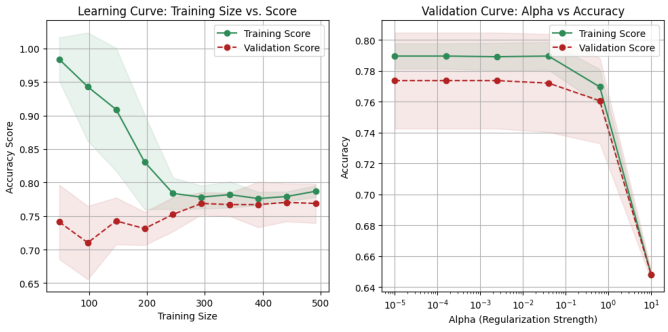

Fig. 2. Neural networks trained with diabetes dataset.
Learning curve: training size vs accuracy (left) and validation curve: alpha vs accuracy (right)

B. Neural Networks: Concrete Strength Dataset

The neural network implementation that achieved the highest accuracy through manual tuning is as follows:

```
MLP = MLPClassifier(
    activation='relu',
    solver='sgd',
    alpha=0.0001,
```

```
tol=0.0001,
learning_rate='adaptive',
hidden_layer_sizes=(12, 12, 12),
max_iter=5000,
random_state=sklearn_random_state)
```

This configuration uses the ReLU activation function, a three-layer architecture with 12 nodes per layer, and the stochastic gradient descent (SGD) optimizer. The learning rate is set to 'adaptive', allowing the model to adjust its learning rate based on the performance. The regularization parameter α is set to 0.0001, with a tolerance of 0.0001, and the model runs for a maximum of 5000 iterations.

The **iterative learning curves** in Figure 3 show higher log-loss values and lower accuracy compared to the diabetes dataset. This can be attributed, in part, to the increased complexity of the multi-class classification problem, which provides more opportunities for misclassifications. The iterations vs. log-loss plot on the left begins with log-loss values exceeding 2.25, gradually stabilizing after approximately 6,000 iterations for the validation set, where the neural network converges. After this point, the validation log-loss starts to increase, signaling the onset of overfitting as the model begins to memorize the training data rather than generalize effectively.

On the right, the iterations vs. accuracy plot shows that accuracy stabilizes at a similar iteration count of around 6,000, but it takes significantly more iterations for the accuracy to level off compared to the diabetes dataset. This further emphasizes the difficulty of the multi-class problem, requiring more iterations for the model to reach convergence.

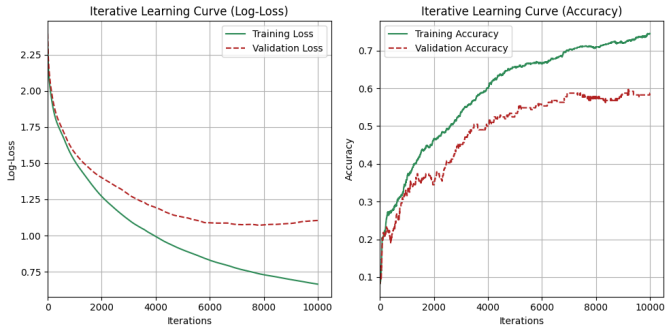

Fig. 3. Neural networks trained with concrete compressive strength dataset. Iterative learning curves: log-loss (left) and accuracy (right)

The **learning curve** in Figure 4 suggests that the neural network model is approaching convergence, but it may benefit from additional data points to ensure that accuracy remains stable for training sizes beyond 650. The **validation curve** for the regularization parameter alpha vs. accuracy shows similar results to Figure 2, indicating that as long as alpha remains below 10^{-2}, the validation accuracy is not significantly affected by this hyperparameter. The maximum accuracy across both graphs is slightly above 60%, which, while lower than the accuracy achieved on the binary classification task of the diabetes dataset, is an expected outcome given the complexity of the multi-class problem. Nevertheless, it represents a reasonable performance.

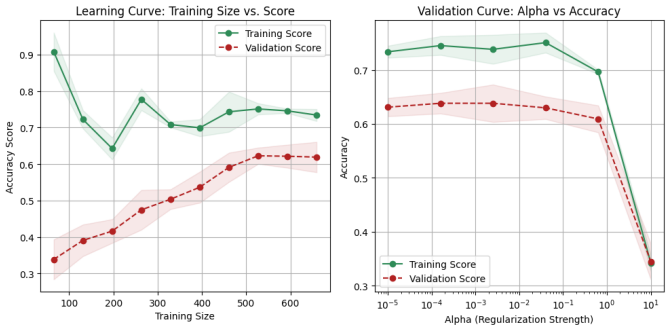

Fig. 4. Neural networks trained with concrete compressive strength dataset. Learning curve: training size vs accuracy (left) and validation curve:alpha vs accuracy (right)

VII. Support Vector Machine (SVM)

Support Vector Machines (SVM) are supervised learning algorithms that classify data by finding the best boundary (hyperplane) that separates different classes [1]. If the data cannot be easily separable, SVM uses techniques (kernels) to handle more complex boundaries. These techniques are useful for high-dimensional data and can prevent overfitting.

Support vector machine algorithms are tested by varying key hyperparameters, including the kernel functions and the regularization parameter C. The results for each dataset are presented in the following subsections.

A. Support Vector Machine: Diabetes Dataset

The Support Vector Machine (SVM) model that achieved the best performance is as follows:

```
svm_model = svm.SVC(
    kernel='linear',
    C=1)
```

This configuration uses a linear kernel for separating the data, with the regularization parameter $C = 1$, which balances the trade-off between maximizing the margin and minimizing classification errors.

Figure 5 presents two curves: the **learning curve** on the left shows how the model's performance improves as more data is added to the SVM model. For the binary classification task on the diabetes dataset, the model converges at a training size of around 300, indicating that beyond this point, adding more data does not significantly improve performance.

On the right, the **validation curve** plots the regularization parameter C against accuracy. Similar to the behavior of the alpha parameter in neural networks, the hyperparameter C has minimal impact on performance as long as C is greater than 1. As noted in the `scikit-learn` documentation, C controls the trade-off between model complexity and generalization by balancing the emphasis on minimizing misclassification errors versus maximizing the margin.

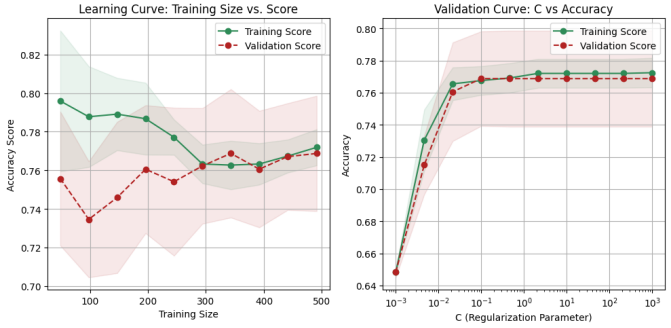

Fig. 5. SVM trained with diabetes dataset.
Learning curve: training size vs accuracy (left) and validation curve: C vs accuracy (right)

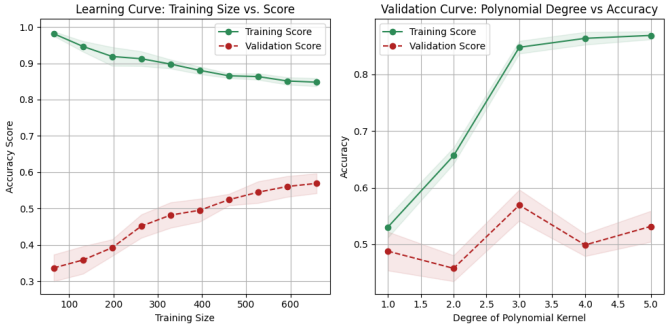

Fig. 6. SVM trained with concrete compressive strength dataset.
Learning curve: training size vs accuracy (left) and validation curve: polynomial degree vs accuracy (right)

B. Support Vector Machine: Concrete Strength Dataset

The Support Vector Machine (SVM) model that achieved the best performance is as follows:

```
svm_model = svm.SVC(
    kernel='poly',
    degree=3,
    C=100)
```

This configuration uses a polynomial kernel with a degree of 3, allowing the model to capture more complex relationships in the data by fitting a curved decision boundary. The regularization parameter $C = 100$ is set to prioritize correct classification over margin maximization, which can lead to a more complex model.

The **learning curve** in Figure 6 shows a clear gap between the training and validation curves, where the training set achieves high accuracy, but the validation set lags with much lower accuracy. This pattern is a strong indicator of overfitting, suggesting that this SVM model could benefit significantly from adding more training data to improve generalization.

The **validation curve**, which explores the hyperparameter of the polynomial kernel degree, which is different from the study of the hyperparamter C shown in Figure 5 for the diabetes dataset. In this plot, the validation accuracy fluctuates between approximately 45% and 55%, while the training accuracy sharply increases to very high levels at degree 3, indicating the onset of overfitting. Since the best accuracy for the validation data is observed around degree 3, this is the selected polynomial degree for this model.

VIII. K-Nearest Neighbors (kNN)

K-Nearest Neighbors (kNN) is a supervised machine learning algorithm that finds the "k" closest data points (neighbors) to a given query point and makes predictions based on their values [1]. The performance of this algorithm depends on the choice of distance metric and the number of neighbors.

k-nearest neighbor algorithms are tested by varying key hyperparameters, including the number of k-neighbors and the weights. The results for each dataset are presented in the following subsections.

A. K-Nearest Neighbors: Diabetes Dataset

The K-Nearest Neighbors (KNN) model that achieved the best performance is as follows:

```
knn_model = neighbors.KNeighborsClassifier(
    n_neighbors=15,
    weights='uniform')
```

This configuration uses 15 neighbors for classification, meaning the model considers the 15 nearest data points when making predictions. The weights are set to 'uniform', indicating that all neighbors contribute equally to the decision, regardless of their distance from the query point.

The **learning curve** in Figure 7 is specific to the kNN algorithm with uniform weights, and its behavior differs somewhat from the other algorithms. In this plot, the accuracy of both the training and validation sets improves steadily as the training size increases, since each neighbor contributes equally to the prediction. As there is no clear plateau, it is likely that increasing the training size further would continue to benefit the model's performance.

The **validation curve** in the same Figure illustrates the effect of the number of neighbors on accuracy. The plot shows that when the number of neighbors exceeds 10, the validation accuracy stabilizes around 75%, with slight oscillations. As the number of neighbors increases, the model's predictions become more like an average of the large cluster surrounding the query point, reducing sensitivity to individual data points.

B. K-Nearest Neighbors: Concrete Strength Dataset

The K-Nearest Neighbors (KNN) model that achieved the best performance is as follows:

```
knn_model = neighbors.KNeighborsClassifier(
    n_neighbors=5,
    weights='distance',
    metric='manhattan')
```

This configuration uses 5 neighbors for classification, meaning the model considers the 5 nearest data points when making predictions. The weights are set to 'distance', meaning that closer neighbors will have a greater influence on the

Fig. 7. kNN trained with diabetes dataset.
Learning curve: training size vs accuracy (left) and validation curve: k-number vs accuracy (right)

classification decision compared to farther neighbors. The distance between points is calculated using the Manhattan distance metric, which sums the absolute differences between the feature values.

The **learning curve** and **validation curve** in Figure 8 are noteworthy due to the use of Manhattan distance for weighting in the kNN model. The training accuracy remains at 100% across varying training sizes and numbers of neighbors because the model effectively "memorizes" the training data. In kNN, each training point serves as its own nearest neighbor, ensuring perfect predictions for the training set.

For the validation accuracy, the learning curve shows a consistent improvement as the training size increases, indicating that the model would likely benefit from additional data to enhance its performance further. The validation curve, however, shows a relatively stable accuracy of around 45% for neighbor values ranging from 2 to 30, with a slight increase in accuracy when $k = 5$.

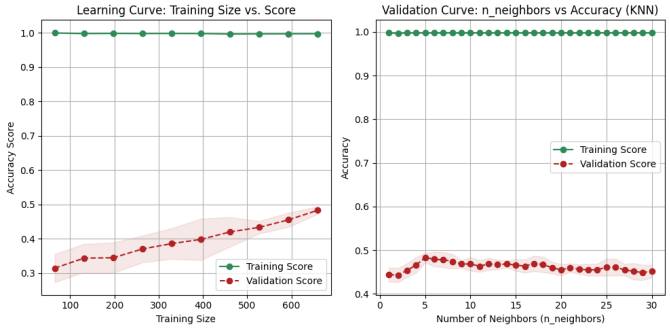

Fig. 8. kNN trained with concrete compressive strength dataset.
Learning curve: training size vs accuracy (left) and validation curve: k-number vs accuracy (right)

IX. Comparison of Learning Algorithms

Table I compares the training and query time complexities of k-Nearest Neighbors (kNN), Support Vector Machines (SVM), and Neural Networks [2], [5], [6]. The complexities are expressed in Big O notation.

TABLE I
Training and Query Time Complexities of kNN, SVM, and Neural Networks

Algorithm	Training Time Complexity	Query Time Complexity
Neural Networks	$O(n \cdot d \cdot h \cdot i)$	$O(d \cdot h)$
SVM (Linear)	$O(n^2 \cdot d)$ to $O(n^3)$	$O(d)$
SVM (Non-linear)	$O(n^2 \cdot d)$ to $O(n^3)$	$O(s \cdot d)$
kNN	$O(1)$	$O(n \cdot d)$

- n represents the number of training samples in the dataset.
- d refers to the number of features or dimensions in each data point.
- h denotes the number of hidden units (or neurons) in the hidden layers of a neural network.
- i represents the number of iterations or epochs in the training process.
- s refers to the number of support vectors used in SVMs when a non-linear kernel is applied.

The experiments in this report confirm the expected computational times for the algorithms implemented. Neural networks took the longest to train (0.7772 s and 7.5365 s for each dataset), as they involve a computationally intensive process of iterating forward and backward propagation through each layer. As shown in Figure 1 and Figure 3, the multi-class classification problem for the concrete compressive strength dataset required over 10 times more iterations to converge compared to the binary classification problem in the diabetes dataset. However, query time was substantially quicker than training time (0.0017 s and 0.0022 s for each dataset), as it only requires a single forward pass through the network. The neural network models were not deep, the diabetes dataset consisted of two layers, while the concrete compressive strength model had three layers.

The SVM algorithm was significantly faster to train (0.0117 s and 0.0894 s for each dataset) and query (0.0019 s and 0.0042 s for each dataset) than the neural networks, largely because the SVM models were simpler. For the diabetes dataset, a linear kernel was used, while the concrete compressive strength dataset was implemented with a third-degree polynomial kernel. Since these datasets were relatively small, the computational cost was low.

The kNN algorithm, being a lazy learning method, did not require a training phase (0.0021 s and 0.0017 s for each dataset). This is why, as shown in Figure 8, when using the kNN model with Manhattan distance as weights, the prediction accuracy for the training set was 100%, as the model simply memorized the training points. The query time for kNN (0.0098 s and 0.0044 s for each dataset), however, depends on the number of training samples and features. In the case of the diabetes and concrete compressive strength datasets, the query time was also computationally manageable due to the simplicity of the data.

Table II summarizes the accuracy obtained after implementing the supervised learning algorithms on the two datasets. It distinguishes between the results of manually tuned hyperpa-

TABLE II
ACCURACY RESULTS IN TEST SET FOR DATABASES AND ML ALGORITHMS

Dataset	Classification	Hyperparameters	NN	SVM	kNN
Diabetes	Binary	Manually tuned	0.779221	0.798701	0.753246
		Grid search	0.775183	0.771958	0.768706
Concrete	Multiclass	Manually tuned	0.567961	0.611650	0.524272
		Grid search	0.634708	0.608004	0.484228

rameters and those found using a grid-search strategy. Interestingly, the accuracy results for manually tuned hyperparameters are comparable to those obtained from grid search, suggesting that when hyperparameter sensitivity is well-understood, manual tuning can yield similar outcomes to grid search. In some cases, the manually tuned results are even slightly higher than those from the grid search. This may be due to the grid search being limited to a specific set of hyperparameters, or the manually tuned prediction having yielded a slightly better result than the grid search's average performance. We got these results because there was a process of understanding the impact of each hyperparameter and supervised learning algorithm, as part of this assignment.

The table also highlights that overall, the performance of the NN, SVM, and kNN models is quite similar in terms of accuracy, with SVM achieving marginally better results in both datasets. However, this does not necessarily mean that SVM is definitively better, as these accuracy values are based on specific train-test splits. It is also notable that kNN exhibits slightly lower accuracy than the other models. The overall difference in accuracy across the models is approximately 6% for the diabetes dataset and 8% for the concrete compressive strength dataset. Note that we determined the "best" algorithm in terms of accuracy, but this could be different if we use different performance metrics, such as precision, recall, or F1 score. A superior performance of the binary classification problem using the diabetes dataset was expected, as a multi-class problem, such as the concrete strength prediction has more bins to predict, and therefore more possibility of being inaccurate.

TABLE III
CONFUSION MATRIX: CONCRETE STRENGTH PREDICTION WITH NN

Bins	0-15	15-25	25-35	35-45	45-55	55-65	65-75	75-85
0-15	78	15	0	0	0	0	0	0
15-25	14	79	33	5	1	0	0	0
25-35	3	16	131	37	5	0	0	0
35-45	0	3	38	131	14	2	0	0
45-55	0	0	5	31	48	17	0	0
55-65	0	0	0	3	17	36	13	0
65-75	0	0	0	0	2	18	19	0
75-85	0	0	0	0	0	1	7	0

An interesting insight arises from the confusion matrices for the concrete compressive strength multi-class predictions. A confusion matrix is a tool used to evaluate classification model performance by comparing actual values to predicted values. Table III shows the confusion matrix for the neural network model predicting concrete compressive strength. Despite the lower accuracy of this multi-class classification task compared to the binary classification of the diabetes dataset, the confusion matrix reveals that, while the exact bin is not always predicted correctly, the model is quite effective at estimating an approximate value. In problems like this, accuracy is not the sole focus—it's often more important to avoid overestimating values, such as the concrete capacity. These results suggest that applying a standard deviation to the predicted value could provide a reliable lower-bound estimate of concrete strength.

To improve the performance of these algorithms, increasing the number of data points would certainly lead to greater certainty in the results. A more comprehensive grid search, exploring a wider range of hyperparameter values, could have potentially uncovered better configurations for each machine learning algorithm. Another factor that could enhance performance is the inclusion of additional features by further exploring the datasets and identifying factors that might have a significant impact on the outcome. Considerable effort was already made to ensure the algorithms performed well by preprocessing the data, scaling the features, using cross-validation, and carefully tuning the hyperparameters manually.

The hypotheses presented in Section III were challenged by the results, highlighting that intuition alone cannot always predict which supervised learning algorithm will perform best. It's always beneficial to implement and compare multiple algorithms, considering the available data and the specific problem. The accuracy for predicting diabetes hovered between 78% and 80%, which is slightly below the anticipated 85%, though still a strong result. For the concrete compressive strength dataset, the models exceeded expectations, achieving accuracies between 53% and 61%, far surpassing the initial estimate of around 40%.

The best-performing algorithm, based on accuracy and the given train-test split, was SVM. This outcome contradicts our original hypothesis, which predicted kNN would perform best for the diabetes dataset and neural networks would excel for the concrete compressive strength dataset.

X. CONCLUSION

In this study, we implemented and evaluated three supervised learning algorithms—neural networks, support vector machines (SVM), and k-nearest neighbors (kNN)—on two distinct datasets. The results demonstrated that choosing the best algorithm for a given problem is not necessarily intuitive, as evidenced by the fact that the algorithms that performed best in our experiments contradicted our initial hypotheses. This emphasizes the importance of implementing and comparing multiple algorithms rather than relying solely on prior experience or intuition. By testing different approaches, we can ensure a more comprehensive evaluation and select the model that truly fits the dataset and task.

Another critical aspect of model performance lies in understanding the hyperparameters of each algorithm. Through manual tuning and grid search, we found that manually tuned models often performed comparably than those tuned via automated search methods. This underscores the need for a strong foundational understanding of the algorithms being used, as well as their hyperparameters, to implement them

effectively. For professionals in computer science, having a good understanding of the basics of neural networks, SVMs, and kNN is key to leveraging these tools in the most effective way possible. Overall, the study reinforces the value of thorough experimentation, careful tuning, and the use of diverse algorithms to achieve robust and reliable machine learning models.

REFERENCES

[1] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[3] Khare, A. D. (2020). *Diabetes dataset* [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset

[4] Yeh, I.-C. (1998). *Concrete compressive strength* [Data set]. UCI Machine Learning Repository. Retrieved from https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength

[5] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[6] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297.