A Comparison of Machine Learning and Deep Learning Algorithms for Diabetes Prediction

Oliver Li
College of Engineering
Boston University
Boston, MA 02215
wsoxl@bu.edu

Bohan Zhang
College of Engineering
Boston University
Boston, MA 02215
bohzhang@bu.edu

Aowei Zhao
College of Engineering
Boston University
Boston, MA 02215
aowei8@bu.edu

Xindong Zhou
College of Engineering
Boston University
Boston, MA 02215
zxd10032@bu.edu

Abstract

Diabetes is a common and serious health condition that requires early and accurate diagnosis to be effectively managed. This study looks at how deep learning and traditional machine learning methods can be used to detect diabetes. Deep learning models, like those used for image and sequence analysis, can find complex patterns in data such as medical records, lab test results, and lifestyle information. Traditional methods, such as decision trees and logistic regression, are simpler and easier to understand but still can be effective. We compare these approaches to see which works better for predicting diabetes, considering factors like accuracy, computational cost, and how easy the results are to explain. The findings show that deep learning often performs better with large and complex data, while traditional methods are quicker and more interpretable. Combining these methods could help create a more reliable and user-friendly system to diagnose diabetes and guide treatment.

Keywords: Machine Learning, Deep Learning

1. Introduction

Diabetes is a prevalent and serious chronic disease that requires timely diagnosis and management to prevent severe health complications. This project aims to leverage Machine Learning (ML) and Deep Learning (DL) techniques to analyze patient data, develop predictive models for early diabetes detection, compare and evaluate their performance based on correct classification rate (CCR). In this project, two kinds of comparisons are designed, different methods on the same dataset and the same methods on different datasets. Through a series of experiments, beginners can gain a deeper understanding of the strengths and weaknesses of machine learning and deep learning. At the same time, they will develop a greater awareness of the importance of parameter selection.

2. Literature Review

2.1. Traditional and Deep Learning Models

In the paper "Diabetes Prediction Using Machine Learning and Explainable AI Techniques," the authors demonstrate the feasibility of comparing traditional machine learning models with deep learning approaches in diabetes prediction. They highlight that traditional models, such as XGBoost, Random Forest, and SVM, can perform comparably to deep learning models, suggesting that traditional algorithms are still valuable in predictive modeling tasks. This comparison provides insights into the strengths and limitations of both approaches, focusing on accuracy and interpretability.

This paper explores how diabetes can be predicted through machine learning algorithms and ontological models. This paper compares six common machine learning algorithms: support vector Machine (SVM), k-Nearest Neighbor (KNN), Artificial Neural Network (ANN), Logistic regression (LR), Naive Bayes (NB), and Decision Tree (DT). Based on the Pima Indian Diabetes Database (PIDD) dataset, the performance of these algorithms is tested by a 10-fold cross-validation and a 66% split mode. Accord-

ing to the test results, SVM and logistic regression have the best overall performance, especially in terms of precision and recall. Naive Bayes performs well on unbalanced datasets with high accuracy and ROC area. KNN performs relatively poorly on most metrics and may not be suitable for high-dimensional or complex medical datasets. In summary, SVM performs well in terms of accuracy, while ANN and KNN perform less well in some cases. In the diabetes prediction task, SVM and logistic regression are the more appropriate choices. Although ANN performs well, it has high computational complexity.

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)	ROC Area
Support Vector Machine (SVM)	77.3	78.5	89.8	83.8	0.72
K-Nearest Neighbors (KNN)	70.2	75.9	79.4	77.6	0.65
Artificial Neural Network (ANN)	75.4	79.8	83.2	81.5	0.79
Naive Bayes (NB)	76.3	80.2	84.4	82.3	0.82
Logistic Regression (LR)	77.2	79.3	88	83.4	0.83
Decision Tree (DT)	73.8	79	81.4	80.2	0.75

Figure 1. Comparison Results

2.2. Wide and Deep Learning Model

In the paper "Wide & Deep Learning for Recommender Systems", the authors present Wide & Deep learning—jointly trained wide linear models and deep neural networks—to combine the benefits of memorization and generalization for recommender systems. The paper suggests that by using both models in combination, the performance of the entire model may increase by 3.9% in app acquisition rate. The Wide component captures explicit feature interactions, enabling memorization of known patterns. The Deep component generalizes better to unseen or less frequent feature combinations. The synergistic combination of these components ensures a balance between memorization (wide) and generalization (deep), improving overall recommendation accuracy. Although the literature paper mainly focuses on recommender systems, their results can also be used in clinical data, where data can be divided into categorical and numerical data sections.

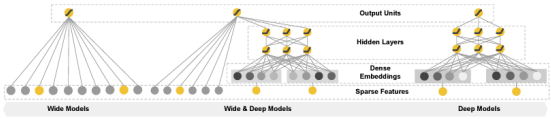

Figure 2. Wide & Deep Learning Model Demonstration

3. Related Work

This part introduces what we have done in this project.

3.1. Dataset Description

We collected three datasets related to diabetes: PIMA Indian (dataset 1), Detailed Health (dataset 2), and LMCH (dataset 3), each varying in size, complexity, and features. Dataset 1 includes 768 samples, each consists of 8 features (pregnancies, glucose, BloodPressure, ...) and 1 label (Yes or No). Dataset 2 has more samples (1789) and features (43). Dataset 3 is very similar to dataset 1. The only difference is that it owns more sample (a larger size dataset).

3.2. Feature Selection

For the feature selection of diabetes data, based on the feature importance analysis of lightgbm, among the 43 features, "Hb1ac" and "fastingbloodsugar" contribute much more to the result than other features. When trained with only these two features, the model will achieve high generalization ability, but when trained with all features, the model is prone to overfitting.

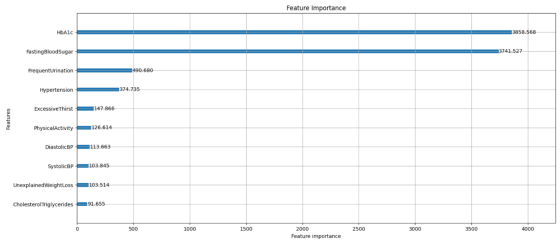

Figure 3. Feature Importance Results

3.3. LightGBM

LightGBM, short for Light Gradient Boosting Machine, is an advanced implementation of the gradient boosting framework. It is an open-source library designed for fast, efficient, and scalable learning, making it highly customizable and convenient for users.

Firstly, a base model is initialized, often using a simple constant value, such as the mean of the target variable. During each training iteration, LightGBM builds new decision trees to minimize the residuals (errors) of the previous model's predictions. Unlike traditional methods, LightGBM grows trees leaf-wise, selecting the leaf with the highest potential reduction in loss to split. This approach enhances accuracy and reduces overfitting in many scenarios.

The algorithm computes the gradient of the loss function with respect to the model's predictions, which is used to determine the optimal adjustment direction. By summing up the predictions of all the trees, the final prediction is obtained, delivering high accuracy with minimal computational cost.

3.4. XGBoost

XGBoost, also known as Extreme Gradient Boosting, is an optimized implementation of the gradient boosting algo-

rithm. It is an open source library, thus it is very convenient for users to import and customize.

Firstly, a base model is initialized, which is a simple model with a constant value, such as the mean of the target variable. For each training iteration, a new decision tree is grown to fit the residuals (errors) of the predictions from the previous mode. The algorithm also computes the gradient of the loss function with respect to the model's predictions. The gradient is used to determine the adjustment direction. By adding up the predictions from each tree, the final prediction is made.

3.5. SVM

SVM, or Support Vector Machine, is a supervised learning algorithm used for classification tasks. It works by finding a hyperplane that best separates data points of different classes. In this implementation, the Radial Basis Function (RBF) kernel is applied, which transforms data into a higher-dimensional space to handle non-linear relationships. The model is trained iteratively on increasing subsets of the data, where critical support vectors define the decision boundary. The RBF kernel is chosen for its flexibility in modeling complex patterns, making it suitable for the given dataset.

3.6. ANN

The number of neurons in the input layer of the ANN model used in this project is consistent with the number of feature dimensions used, the number of neurons in the hidden layer is fixed to 16, and the activation function is ReLU. The output layer is 1 neuron, and sigmoid is used as the activation function for binary classification

3.7. DNN

The Deep Neural Network (DNN) implemented in this project is designed for binary classification. It consists of multiple fully connected layers and uses the ReLU activation function for hidden layers to introduce non-linearity, while the final output layer uses the sigmoid activation function for binary classification. For the input layer, it takes features from the dataset. Then three dense layers with 64, 32, and 16 neurons respectively, are added as hidden layer. All of them use ReLU activation. Finally, a single neuron with sigmoid activation for predicting the probability of the positive class works as the output layer. We choose Adam as the optimizer since it is efficient in training neural networks. Regularization is also applied to get better results.

3.8. Wide and Deep Learning

Wide and Deep Learning is a hybrid model architecture that combines the strengths of wide linear models and deep neural networks. It is designed to handle both memorization and generalization, making it especially effective for tasks such as recommendation systems and search ranking. This framework is open-source and provides flexibility for customization based on specific use cases.

The wide component is a linear model that captures feature interactions in a memorization style. It works well with categorical features and ensures the model learns specific patterns that are explicitly provided in the data.

The deep component is a neural network that learns complex patterns and representations from raw input features. It generalizes well by transforming inputs through multiple non-linear layers, capturing hidden structures and feature interactions that are not explicitly defined.

During training, the wide and deep components are jointly optimized. Their outputs are combined, typically through summation, to generate the final prediction. This integration allows the model to leverage both the explicit feature interactions from the wide component and the implicit patterns captured by the deep component, resulting in a robust predictive model. See the Appendix for more information.

4. Results

4.1. Different Algorithms on the Same Dataset

We test algorithms on three datasets and get results as below.

Algorithm	Precision (mean)	Precision (max)	Recall (mean)	Recall (max)	F1-score (mean)	F1-score (max)	Run Time (sec)
XGBC	0.51	0.72	0.55	0.81	0.62	0.77	2.10
LightGBM	0.32	0.65	0.5	1	0.51	0.79	1.66
SVM	0.70	0.74	0.40	0.44	0.51	0.54	1.14
DNN	0.70	0.79	0.70	0.77	0.70	0.78	4.39
ANN	0.74	0.81	0.74	0.82	0.74	0.81	4.10

Figure 4. Dataset-1

Algorithm	Precision (mean)	Precision (max)	Recall (mean)	Recall (max)	F1-score (mean)	F1-score (max)	Run Time (sec)
XGBC	0.88	0.91	0.87	0.92	0.89	0.91	5.55
LightGBM	0.93	0.95	0.91	0.97	0.92	0.94	3.10
SVM	0.64	0.66	0.56	0.62	0.60	0.64	4.00
DNN	0.81	0.84	0.80	0.87	0.81	0.86	7.94
ANN 2 features	0.90	0.93	0.88	0.96	0.89	0.92	7.10
ANN all 43 Features	0.80	0.85	0.80	0.85	0.80	0.85	7.12
Wide and Deep Learn	0.84	0.88	0.79	0.94	0.80	0.86	13.83

Figure 5. Dataset-2

As these figures shown, LightGBM and XGBoost perform better on larger, more complex datasets. SVM lacks adaptability to dataset complexity. DNN and ANN perform similarly.

Algorithm	Precision (mean)	Precision (max)	Recall (mean)	Recall (max)	F1-score (mean)	F1-score (max)	Run Time (sec)
XGBC	0.85	0.97	0.82	0.96	0.95	0.96	7.66
LightGBM	0.97	0.98	0.97	0.98	0.97	0.97	8.59
SVM	0.78	0.81	0.47	0.47	0.58	0.59	5.73
DNN	0.98	0.99	0.98	0.99	9.98	0.99	7.02
ANN	0.79	0.83	0.78	0.86	0.78	0.84	6.61

Figure 6. Dataset-3

4.2. SVM Kernal Selection

In the table, 'NaN' indicates that a result could not be obtained within the specified time limit.

Dataset/kernel	Linear precision(max)	Poly precision(max)	Rbf precision(max)	Sigmoid precision(max)
1	0.66	0.74	0.68	0.23
2	Nan	0.66	0.64	0.57
3	Nan	0.82	0.78	0.13

Figure 7. SVM max precision by different kernel method

SVM's performance varies across the three datasets due to differences in sample size and feature complexity. The polynomial kernel was selected for its ability to capture complex, non-linear relationships, leading to the highest precision among tested models.

5. Conclusion

We have tested six algorithms and three datasets to compare the performance of ML and DL methods on diabetes prediction. When dealing with small datasets, ML models can still perform adequately, but there are some key considerations to enhance their effectiveness. One important step is screening the feature dimensions, particularly those that show high correlation with the results, before training the model. This helps in reducing overfitting, as it prevents the model from learning patterns that are simply noise or artifacts of the dataset rather than genuine relationships. However, due to the relatively low quality of the available data, DNN may not outperform simpler architectures like ANN. This is because DNN typically require larger datasets to capture complex patterns effectively, and with limited data, their performance may degrade due to overfitting. Furthermore, the collection of diverse and relevant features is crucial to improving model performance. By carefully selecting the features and model complexity, better performance can be achieved, even in situations with limited data quality.

6. Description of Individual Efforts

For report responsibility, please see table 1, the distribution in Related Work is consistent with the coding part

For coding responsibility, please see table 2

Part Name	Contributor
Abstract	Oliver
Literature Review	Aowei, Oliver
Related Work	Everyone
Results	Bohan, Xindong
Conclusion	Aowei
Description	Aowei
Appendix	Everyone

Table 1. Report Distribution

Name	Duty
Aowei	XGBoost, DNN
Bohan	SVM, Code Integration
Oliver	LightGBM, Wide and Deep Learning
Xindong	ANN, Dataset Analysis

Table 2. Code Distribution

For literature survey, everyone makes an equally important contribution.

For algorithm analysis, the distribution is consistent with the coding part.

7. Appendix

7.1. Pseudo-code

Algorithm 1 LightGBM

Require: Training data X, target labels y, hyperparameters: learning rate η, max depth d, boosting rounds T, subset size scaling k.

Ensure: Trained LightGBM model and evaluation results.

Initialize parameters:

$$\text{params} \leftarrow \{\text{objective: 'binary', metric: 'auc'}, \eta, d\}$$

Initialize empty results list: results $\leftarrow []$

for iteration i in range(50, 101) **do**

Compute subset size:

$$\text{subset\_size} \leftarrow \max(1, \lfloor \text{len}(X_{\text{train}}) \cdot \frac{i}{100} \rfloor)$$

Prepare subset:

$$X_{\text{subset}}, y_{\text{subset}} \leftarrow X_{\text{train}}[:\text{subset\_size}], y_{\text{train}}[:\text{subset\_size}]$$

Construct LightGBM dataset:

$$\text{train\_data} \leftarrow lgb.Dataset(X_{\text{subset}}, label = y_{\text{subset}})$$

Train LightGBM model:

$$\text{model} \leftarrow lgb.train(\text{params}, \text{train\_data}, num\_boost\_round = T)$$

Predict on test set:

$$\hat{y} \leftarrow \text{model.predict}(X_{\text{test}})$$

Compute accuracy:

$$\text{Accuracy} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \Big(I(\hat{y}_i \geq 0.5 \text{ and } y_{\text{test},i} = 1) + I(\hat{y}_i < 0.5 \text{ and } y_{\text{test},i} = 0) \Big)$$

Append accuracy to results: results.append(Accuracy)

end forreturn Trained LightGBM model and results.

Algorithm 2 XGBoost

Require: Training data $\{X, y\}$, parameters: max_depth, learning_rate, λ, γ, number of trees T

Ensure: Final model F_T

Initialize model: $F_0 \leftarrow \text{mean}(y)$

for $t = 1$ to T **do**

Compute residuals: $r_i \leftarrow y_i - F_{t-1}(X_i)$ for $i = 1, 2, \ldots, n$

Train a new decision tree $Tree_t$ to fit r_i using X

for each leaf l in $Tree_t$ **do**

Compute optimal weight for leaf l:

$$w_l \leftarrow -\frac{\sum_{i \in l} r_i}{\sum_{i \in l} (\text{hessian}_i) + \lambda}$$

end for

Update the model:

$$F_t(X) \leftarrow F_{t-1}(X) + \eta \cdot Tree_t(X)$$

Apply regularization:

$$\text{Regularization} = \gamma \cdot \text{num(leaves)} + \lambda \cdot \text{sum(leaf weights)}^2$$

Update the objective:

$$\text{Objective} = \text{Loss}(F_t, y) + \text{Regularization}$$

end for

return F_T

Algorithm 3 SVM with Kernel Selection

Require: Training data X, target labels y, hyperparameters: kernel type (`rbf`), random state, iterations, subset size.

Ensure: Trained SVM model.

Initialize model: $SVM \leftarrow SVC(\text{kernel='rbf', probability=True, random\_state=42})$

for iteration i in range(50, 101) **do**

Compute subset size:

$$\text{subset\_size} = \max(1, \lfloor \text{len}(X_{\text{train}}) \cdot \frac{i}{100} \rfloor)$$

Fit model on subset: $SVM.\text{fit}(X_{\text{train}}[:\text{subset\_size}], y_{\text{train}}[:\text{subset\_size}])$

Predict on test set: $\hat{y} = SVM.\text{predict}(X_{\text{test}})$

Compute accuracy: Accuracy $= \frac{1}{n} \sum_{i=1}^{n} I(\hat{y}_i = y_{\text{test},i})$

Append accuracy to results: results[S̈VM_kernel¨].append(Accuracy)

end forreturn Trained SVM model and results.

Algorithm 4 Artificial Neural Network (ANN)

Require: Training data X, target labels y, network parameters $\{W, b\}$, learning rate η, batch size b, number of epochs E.

Ensure: Trained ANN model parameters $\{W, b\}$.

Initialize network parameters: $\{W, b\} \leftarrow$ random initialization.

for epoch e in range(1, E) **do**

  for batch $(X_{\text{batch}}, y_{\text{batch}})$ in Batches(X, y, b) **do**

    Perform forward propagation:

$$A_0 \leftarrow X_{\text{batch}}$$

    for each layer l in network (1 to L) **do**

$$Z_l \leftarrow W_l \cdot A_{l-1} + b_l$$

$$A_l \leftarrow \sigma(Z_l) \quad \text{(activation function)}$$

    end for

$$\hat{y} \leftarrow A_L \quad \text{(output of final layer)}$$

    Compute loss (binary cross-entropy):

$$L \leftarrow -\frac{1}{b} \sum_{i=1}^{b} \Big(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \Big)$$

    Perform backward propagation:

$$dA_L \leftarrow \hat{y} - y_{\text{batch}}$$

    for layer l in L to 1 (backward order) **do**

$$dZ_l \leftarrow dA_l \cdot \sigma'(Z_l)$$

$$dW_l \leftarrow \frac{1}{b} \cdot (dZ_l \cdot A_{l-1}^T)$$

$$db_l \leftarrow \frac{1}{b} \cdot \text{sum}(dZ_l)$$

$$dA_{l-1} \leftarrow W_l^T \cdot dZ_l$$

    end for

    Update parameters:

$$W_l \leftarrow W_l - \eta \cdot dW_l, \quad b_l \leftarrow b_l - \eta \cdot db_l$$

  end for

end forreturn Trained ANN model parameters $\{W, b\}$.

Algorithm 5 Deep Neural Network (DNN)

Require: Feature matrix X, target labels y, hyperparameters: epochs, batch size, validation split

Ensure: Trained DNN model

Normalize X using standard scaler

Split data into training set $(X_{\text{train}}, y_{\text{train}})$ and test set $(X_{\text{test}}, y_{\text{test}})$

Initialize a sequential mode, add layers and set parameters

Train the model on $(X_{\text{train}}, y_{\text{train}})$

Evaluate the model on $(X_{\text{test}}, y_{\text{test}})$

Get results if required

return DNN model

Algorithm 6 Wide and Deep Learning

Require: Training data $X_{\text{wide}}, X_{\text{deep}}$, target labels y, wide component weights W_{wide}, deep network parameters θ_{deep}, learning rate η, batch size b, number of epochs E.

Ensure: Trained Wide and Deep model parameters.

Initialize wide component weights: $W_{\text{wide}} \leftarrow$ random initialization

Initialize deep network parameters: $\theta_{\text{deep}} \leftarrow$ random initialization

for epoch e in range(1, E) **do**

  for batch $(X_{\text{wide}}, X_{\text{deep}}, y)$ in Batches$(X_{\text{wide}}, X_{\text{deep}}, y, b)$ **do**

    Compute wide component output:

$$y_{\text{wide}} \leftarrow X_{\text{wide}} \cdot W_{\text{wide}}$$

    Compute deep component output:

$$h_0 \leftarrow X_{\text{deep}}$$

    for each layer l in deep network **do**

$$h_l \leftarrow \sigma(W_l \cdot h_{l-1} + b_l)$$

    end for

$$y_{\text{deep}} \leftarrow h_L \quad \text{(output of final layer)}$$

    Combine wide and deep outputs:

$$y_{\text{pred}} \leftarrow \sigma(y_{\text{wide}} + y_{\text{deep}})$$

    Compute loss (e.g., binary cross-entropy):

$$L \leftarrow -\frac{1}{b} \sum_{i=1}^{b} \Big(y_i \cdot \log(y_{\text{pred},i}) + (1 - y_i) \cdot \log(1 - y_{\text{pred},i}) \Big)$$

    Backpropagate to update parameters:

$$W_{\text{wide}} \leftarrow W_{\text{wide}} - \eta \cdot \nabla_{W_{\text{wide}}} L$$

$$\theta_{\text{deep}} \leftarrow \theta_{\text{deep}} - \eta \cdot \nabla_{\theta_{\text{deep}}} L$$

  end for

end forreturn Trained model parameters $W_{\text{wide}}, \theta_{\text{deep}}$

References

[1] H. E. Massari, Z. Sabouri, S. Mhammedi, and N. Gherabi, "Diabetes prediction using machine learning algorithms and ontology," *Journal of ICT Standardization*, vol. 10, no. 2, pp. 319–338, 2022.

[2] M. R. Islam, A. Haque, H. Iqbal, and M. Kamruzzaman, "Diabetes prediction using machine learning and explainable ai techniques," *Health Technology Letters*, vol. 10, no. 1, p. e10107388, 2023.

[3] O. Iparraguirre-Villanueva *et al.*, "Application of machine learning models for early detection and accurate classification of type 2 diabetes," *Diagnostics*, vol. 13, no. 14, p. 2383, 2023.

[4] L. Jiang, Z. Yang, D. Wang, *et al.*, "Diabetes prediction model for unbalanced community follow-up data set based on optimal feature selection and scorecard," *Digital Health*, vol. 10, 2024.

[5] S.-L. Lin, "Application of machine learning to a medium gaussian support vector machine in the diagnosis of motor bearing faults," *Electronics*, vol. 10, no. 18, p. 2266, 2021.

[6] S. Modak and V. Jha, "Diabetes prediction model using machine learning techniques," *Multimedia Tools and Applications*, vol. 83, pp. 1–27, 2023.

[7] M. M. Owess, A. Y. Owda, M. Owda, and S. Massad, "Supervised machine learning-based models for predicting raised blood sugar," *International Journal of Environmental Research and Public Health*, vol. 21, no. 7, p. 840, 2024.

[8] V. Rawat and Suryakant, "A classification system for diabetic patients with machine learning techniques," *International Journal of Mathematical, Engineering and Management Sciences*, vol. 4, no. 3, pp. 729–744, 2019.

[9] W. Yu, T. Liu, R. Valdez, *et al.*, "Application of support vector machine modeling for prediction of common diseases: the case of diabetes and pre-diabetes," *BMC Medical Informatics and Decision Making*, vol. 10, p. 16, 2010.

[10] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," *arXiv preprint arXiv:1606.07792*, 2016.