Victoria De Cun
Asif Abbas
Moein Afrouzmehr
Fidel Gutierrez
Indronil Bhattacharjee

CS508 - Intro to Data Mining
Final Report

25 November 2023

Diabetes Predictive Modeling: Leveraging Machine Learning for Early Detection

Navigating the Diabetes Epidemic: Insights from Recent Studies and Surveys

Diabetes remains a pressing global health issue, with recent studies underscoring its widespread impact and the evolving landscape of treatment and management. According to the National Diabetes Statistics Report by the CDC, the prevalence of diabetes in the United States is significant, affecting 11.6% of the total population, or 38.4 million people of all ages. Among adults aged 18 years or older, the prevalence rises to 14.7%, totaling 38.1 million individuals. This data highlights the extensive reach of the disease and the critical need for effective management strategies and public health interventions Centers for Disease Control and Prevention (2021).

The year 2023 has seen remarkable advancements in the treatment of diabetes, particularly in the realm of insulin therapies. New developments in insulin formulations and the approval of biosimilars, such as Admelog (insulin lispro), have made diabetes management more affordable and accessible. These advancements represent a significant step forward in ensuring that patients have access to essential diabetes treatments. This progress is crucial given the high prevalence of diabetes and its impact on healthcare systems worldwide *Trailblazing Discoveries: The Top 5 Diabetes Research Breakthroughs of 2023* (2023).

In response to these challenges, there have been updates to the Standards of Care in Diabetes in 2023, which provide guidance for primary care providers. These updates include crucial new information, especially in the areas of classification and diagnosis of diabetes. This is indicative of the dynamic nature of diabetes care, where continuous research and updated clinical guidelines are vital for effective management. Additionally, the Global Diabetes Compact by the WHO unites stakeholders around the goals of reducing diabetes risk and ensuring equitable access to comprehensive, affordable care, and prevention. This initiative reflects the global commitment to combating diabetes and the recognition of its broad socio-economic impact. Furthermore, the focus on specialized care aspects, like school-based diabetes care for children with type 1 diabetes, emphasizes the need for tailored approaches in different settings and age groups, addressing unique challenges and facilitating effective diabetes management (“Improving health outcomes of people with diabetes: target setting for the Global Diabetes Compact”, 2023; “School-Based Diabetes Care: A National Survey of U.S. Providers”, 2023; “Standards of Care in Diabetes—2023 Abridged for Primary Care Providers”, 2023).

Objective

The chief objective of this project is to employ a variety of machine learning classifiers on a dataset comprising individuals both with and without diabetes, with the aim of developing a sturdy predictive model. This model will utilize a diverse array of features, spanning clinical and non-clinical, to ascertain an individual’s risk of developing diabetes. Through harnessing the predictive capabilities of machine learning algorithms, we aspire to improve existing screening methods and proactively pinpoint individuals at elevated risk.

Motivation

1. **Early Detection and Intervention**: A majority of complications arising from diabetes are a result of delayed diagnosis and treatment. By predicting susceptibility, one can instigate early interventions thereby reducing the risk of complications.
2. **Enhancing Existing Diagnostic Tools**: While traditional diagnostic tools focus on overt symptoms and clinical markers, a machine learning model can consider a wide array of features, potentially uncovering novel indicators of diabetes risk.
3. **Holistic Understanding of Contributing Factors**: By evaluating the significance of various features in predicting diabetes, we can gain insights into less-understood factors that contribute to its onset.

Dataset

- **Source** - The dataset was obtained from Kaggle.
- **Category** - This dataset originates from the healthcare domain. The class label is a binary attribute indicating whether a patient has diabetes. The attributes encompass various patient-specific details, including age, BMI, glucose levels, cholesterol levels, among others.
- **Features and size** - The dataset comprises 18 numerical features, and 4304 instances. All inter-feature correlations are below 0.35. Consequently, no attributes are excluded from the analysis.
- **Data preprocessing and statistical analysis** - The statistical analysis and the data preprocessing is explained in detail in an accompanying document.

Classification

Classification is a supervised data mining problem where the goal is to predict categorical labels of instances, based on past data. Classification problems fall under supervised learning, and thus requires some training data. Each instance in the training dataset belongs to a specific class. The dataset used for this project contains a list of patients and some patient-specific attributes such as their age, BMI, glucose levels, cholesterol levels etc. The class label is a binary attribute that indicates whether the patient has diabetes.
The classification algorithm functions by studying the relationship between the attributes of the training set and their corresponding class labels. Having studied the training set, the objective of the algorithm is to predict the most likely classes for previously unseen instances. Common algorithms for classification include Decision Trees, Support Vector Machines (SVM), Logistic Regression, Random Forests, and Neural Networks. The algorithms used in this project are described in detail in the later sections.
The classification algorithms are evaluated by two metrics - their running time and their accuracy. The results of all the algorithms used in this study are described in the later sections.

Algorithms Used

Decision Tree

Decision Trees are one of the simplest classification algorithms, preferred for their ease of use. A Decision Tree in much like a real tree, starting with a root node that branches into additional nodes until we reach an output (also called a leaf node).
Each node can be thought of as representing a test, with each branch from the node representing the outcome of that test. The split at each node is determined by a specific criterion. In this project, we use the **Gini**

impurity as our splitting criterion. The Gini coefficient is a measure of the probability that a randomly chosen instance from a subset will be incorrectly classified. Therefore it follows that the Gini impurity has to be as low as possible. A Gini impurity of 0 implies all of the instances in the subset belong to the same class, implying a perfect split at the node. In practice, the decision tree evaluates the Gini impurity for each potential split of the dataset and picks the split that minimises the Gini impurity.
This process is continued until some stopping conditions are met. Without a limit on the depth of the tree, a decision tree can grow until it has perfectly classified all training data, which often leads to complex models that do not generalize well to new, unseen data. In practice, Decision Trees are truncated using the *max_depth* parameter, which limits the maximum depth of the tree. In our project, we set **max_depth = 6**, which gives us the best results.

Results

- **Running Time**: $0.015\,\mathrm{s}$
- **Accuracy**: 95.66%

Random Forest

The Random Forest classifier is known for its high accuracy and robustness. It leverages the power of multiple decision trees to obtain a more accurate output.
The algorithm works by creating a user-specified number of trees, with each tree trained on a subsample of the training data. The subsamples are bootstrapped, meaning the sampling is with replacement. For each tree, the splitting at each node is based off a random subsample of the total features. This further increases the diversity among the trees.
A Random Forest assigns a class label to an input instance by taking the majority vote from all the trees. Since the majority votes are avergaed, the individual trees can be grown deep and perhaps even fully, without concern for overfitting. Overfitting in decision trees occurs when a tree is overly complex and captures noise in the data. By averaging predictions from multiple trees, Random Forests mitigate this effect. The errors of individual trees can cancel each other out to some extent, leading to more robust overall predictions. On the flip side, this increases model complexity and running time, especially as the number of trees increases. In our example, we stuck with the default **n_estimators = 100**.

Results

- **Running Time**: $0.82\,\mathrm{s}$
- **Accuracy**: 96%

k-Nearest Neighbours

The kNN algorithm works on the simple premise of finding the k nearest neighbours to a given instance, and predicting the class label as the majority vote among the neighbours. This makes the distance metric and the choice of k vital hyperparameters.
kNN algorithms struggle with scalability, as it requires calculating and comparing distances for each new point, and dimensionality, due to the increase in distance computation complexity. Thus, the choice of a reasonable k is paramount. A k that is too small, like 2 or 3, can lead to overfitting. If the data is noisy, a slightly larger k can help to smooth out the noise, making the algorithm less sensitive to outliers. However, too large a value might lead to underfitting. We chose **k=5** as it offered a good balance between underfitting and overfitting. Additionally, an odd value for k prevents any ties when taking the majority vote.
We chose the **Euclidean metric** as our distance metric. The Euclidean distance is the straight-line distance

between two points in Euclidean space. The Euclidean distance is intuitive and generalises extremely well to higher dimensions.

Results

- **Running Time**: $0.004\,\mathrm{s}$
- **Accuracy:** 94.7%

Logistic Regression

Logistic Regression models work by using a the sigmoid function to to model the probability that a given input belongs to a particular class, based on its features X_1 to X_n. The sigmoid function is defined as:

$$\mathrm{P(Y)} = \frac{1}{1 + \exp\left(-\left(\beta_0 + \sum_{\mathrm{i}=1}^{\mathrm{n}} \beta_\mathrm{i} \mathrm{X}_\mathrm{i}\right)\right)}$$

where the coefficients β_i are estimated using maximum likelihood estimation. The goal is to find the set of coefficients that maximize the likelihood of the observed data.
The sigmoid function outputs a value between 0 and 1, which is interpreted as the probability a given input belongs to a particular class. Logistic Regression is primarily used to predict the probability of a binary outcome. A threshold value, typically 0.5, is used to classify predictions. If $\mathrm{P(Y)} > 0.5$, the output is classified as class 1; otherwise, it's class 0.
Due to the non-linear nature of the logistic function, we have rely on iterative methods to solve the maximum likelihood problem. We chose **solver = 'lbfgs'**, which stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno, an optimizer in the family of quasi-Newton methods. We set the maximum number of iterations taken for the solvers to converge at **max_iter = 10000**. We prevent overfitting by penalizing large values for β. This is done using the inverse regularisation parameter C. In our project, we fixed **C=100**.

Results

- **Running Time**: $0.03\,\mathrm{s}$
- **Accuracy**: 96%

Support Vector Classifier

The SVC works on the very simple core concept of finding the hyperplane that best separates different classes in feature space. The best hyperplane maximises the distance between itself and the support vector from each class i.e. the instance from each class closest to the hyperplane.
In real-world data, patterns are often too complex to be separated by a simple line or curve in their original form. Kernels make it possible to handle these complexities. A kernel function is a mathematical trick to map scattered points into a new higher dimensional space where they can be easily separated. The efficiency of kernel functions lies in the fact that they do this transformation without actually without explicitly computing the coordinates in that space, which can be computationally expensive.
The kernel function is a crucial part of SVC, transforming the input data into a higher-dimensional space where it is easier to find a linear separation. Different kernel functions can be used, each leading to different decision boundaries. We used both a **Linear** and a **Radial Basis Function (RBF)** Kernel.
A linear SVC uses the simplest kernel, the linear kernel. In other words, no transformation is applied and the SVC attempts to fit a straight hyperplane to the data. Linear SVCs tend to be faster and require less computational resources, making them efficient for large datasets with a large number of features. But they can lead to oversimplifications if the data cannot be linearly separated.

Results

- **Running Time**: $0.13\,\mathrm{s}$
- **Accuracy**: 95.6%

A Radial Basis Function (RBF) SVC uses the Radial Basis Function kernel to transform the input space. This kernel adds more complexity by creating non-linear boundaries, allowing the hyperplane to capture more complex relationships. But this also adds additional hyperparameters, like the gamma parameter, that need to be carefully tuned. The gamma parameter shapes the decision boundary - a higher gamma makes the boundary increasingly sensitive to each point and a lower gamma results in a much smoother boundary. Adjusting gamma is a balancing act between making the model sensitive enough to capture the nuances in the data and not so sensitive that it starts modeling noise.

Results

- **Running Time**: $0.10\,\mathrm{s}$
- **Accuracy**: 94.2%

Gradient Boosting

Like a Random Forest, a GB works by combining the output of multiple decision trees. In contrast to a Random Forest where all the trees are added in the beginning, a GB starts with a trivial model with new trees being added to correct the residual errors made by the previous models. Each tree in the series is typically assigned a learning rate, which scales the contribution of each tree. A smaller learning rate requires more trees but can often lead to a more accurate model. This process continues until a specified number of trees is reached. We defined this number to be **n_estimators = 200**.
GB, like Random Forest, creates extremely accurate trees but building trees sequentially can be computationally expensive and time-consuming, especially with large datasets.

Results

- **Running Time**: $2.22\,\mathrm{s}$
- **Accuracy**: 95%

Neural Networks

Perceptron

A Perceptron is the simplest neural network. It takes the set of features from the data and assigns a weight to each feature. Each input instance is then multiplied by the appropriate weights and summed. The weighted sum is passed to an activation function, which gives an output. The output is then compared to the true label for the input to determine the error. The weights are subsequently adjusted, based on the error. This is typically done using a method called the Perceptron Learning Rule, which involves adding a fraction of the input value to the weight if the Perceptron misclassifies an instance. The Perceptron is designed for binary classification tasks, which makes it highly suited for our classification problem.

Results

- **Running Time**: $0.003\,\mathrm{s}$
- **Accuracy**: 93%

Artificial Neural Network

An ANN consists of layers of interconnected nodes or neurons. There are three types of layers: input, hidden, and output. Each neuron in one layer is connected to neurons in the next layer through weighted connections. The input layer receives the input data and passes them to the next layer, often the hidden layers. Hidden layers perform computations on the inputs received. These layers are not exposed to the external environment and can have complex structures in deeper networks. Each neuron in these layers applies a weighted sum on its inputs and then an activation function to the sum. The activation function introduces non-linearity, allowing the network to model complex relationships.
In the project, we use a four-layered neural network, with 3 hidden layers. We use a Sequential model, which is a linear stack of layers. This is one of the simplest types of ANN architecture. The input layer has **128 neurons**, with a Rectified Linear Unit (ReLU) activation function. The first hidden layer has **64 neurons**, with ReLU being used again as the activation fucntion. The second hidden layer layer has **32 neurons**, and again, it uses the ReLU activation function. The third hidden layer also functions as the output layer for this network. It contains a **single neurons** as it is meant for a binary classification task. The sigmoid activation function is used because it is appropriate for binary classification, as it squashes the output between 0 and 1, representing a probability. The learning rate is set to **learning_rate = 0.0001**, which controls how much the model's weights are updated during training.

Results

- **Running Time**: $26.10\,\mathrm{s}$
- **Accuracy**: 95%

Conclusion

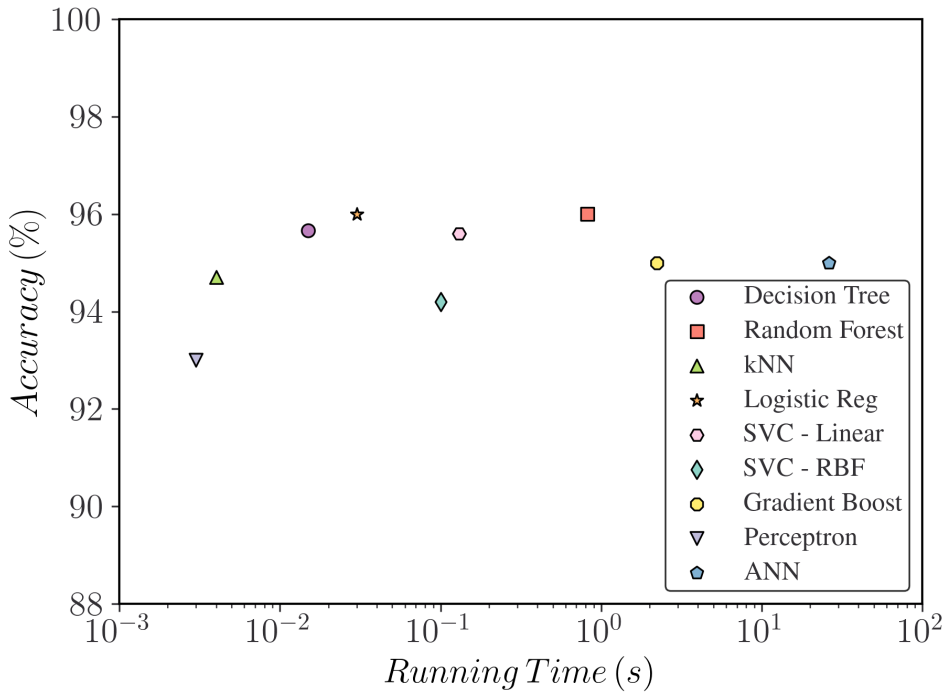

Figure 1: Comparison of the classification algorithms used. Note that the x-axis is in log scale.

Figure 1 shows a comparison of all the classification algorithms used in this work. The accuracy of the algorithm (y-axis) is plotted against the running time in seconds (on the x-axis). Note that the x-axis is in

log scale. In terms of running time, Perceptron and kNN are clear winners while RF and Logistic Regression have the highest accuracy. It is interesting to note that rather counter-intuitively, accuracy does not scale with running time.

References

Centers for Disease Control and Prevention. (2021). *National diabetes statistics report* (Tech. Rep.). CDC. Retrieved from https://www.cdc.gov/diabetes/data/statistics-report/index.html (Accessed: 2023-11-25)

Improving health outcomes of people with diabetes: target setting for the global diabetes compact. (2023). *The Lancet*. Retrieved from https://www.thelancet.com (Accessed: 2023-11-25)

School-based diabetes care: A national survey of u.s. providers. (2023). *Hindawi*. Retrieved from https://www.hindawi.com (Accessed: 2023-11-25)

Standards of care in diabetes—2023 abridged for primary care providers. (2023). *Diabetes Care*, *46*(Suppl. 1), S19–S40. Retrieved from https://diabetesjournals.org/care/article/46/Supplement_1/S19/138770/2023-Standards-of-Care-Update (Accessed: 2023-11-25)

Trailblazing discoveries: The top 5 diabetes research breakthroughs of 2023. (2023). Retrieved from https://www.news-medical.net (Accessed: 2023-11-25)