Group 5
Machine Learning Methods and Evaluation for Early Diabetes Detection and Health Prediction

Marlene Arredondo, Aleksa Kostic, Kevin Pham
Data Science
University of California, San Diego

1 Abstract

For this capstone project, we replicated part of Rich Caruana and Alexandru Niculescu-Mizil's paper "An Empirical Comparison of Supervised Learning Algorithms."[2] This paper compared the performances of several supervised learning algorithms, such as Support-Vector Machines, Naive Bayes, and Decision Trees on datasets using different metrics including accuracy, precision, and F1 score. We proceeded with the performed algorithms such as Naive Bayes, Decision Trees, Random Forest, Radial Basis Function (RBF) Trees, RBF Neural Networks, K-Nearest Neighbors, Logistic Regression, and Support Vector Machines. With these algorithms, we compared their final metrics such as accuracy, precision, and F1 score. After comparing these metrics across all combinations of models and different validation methods such as cross validation and k-fold validation, we found that whether or not we used PCA, the model with the best set of combined metrics (accuracy, precision, and F1 score) was the RBF neural network. However, as a standalone metric, without PCA, KNNs performed best in just accuracy, and with PCA, SVMs performed best in just accuracy.

2 Introduction

In 2021, a staggering 29.7 million in the United States were diagnosed with diabetes, representing 8.9% of the population. Diabetes imposes a substantial burden on individuals' health, increasing the risk of serious complications such as heart attacks and strokes. Machine learning emerges as a tool for uncovering subtle correlations and identifying potential health issues, including diabetes. It provides a broader understanding of health patterns, facilitating more proactive and personalized healthcare interventions.

Previous work has solved this problem by evaluating performance across multiple modes of supervised learning techniques from as simple as Naive Bayes to as complicated as RBF neural networks. Their methodology did not involve preprocessing the data, nor utilizing feature extraction techniques. Conclusions were generally drawn with a focus on accuracy over other performance metrics. Supervised learning algorithms perform quite well when the output is binary as performance can be evaluated using TP, TN, FP, and FN alone. Some methods consistently outperform others such as neural networks. These results could be generalized to prediction in medicine utilizing classification machine learning algorithms.

3 Related Work

Of the related literature, we found that many explored similar algorithms and metrics like our own. The first paper we researched was titled "Comparing different supervised machine learning algorithms for disease prediction" and was extremely similar in methodology and intent as our own replication. This paper combed through various scientific papers, all of which are regarding supervised machine learning algorithms, and compared them based on a set of metrics. One thing to note that we believe was extremely well done and meticulous was their methodology in identifying which articles to examine and compare.[13] The second paper we examined was titled "Supervised Machine Learning Algorithms: Classification and Comparison." Similar to the

previous paper, this paper also explores seven different supervised machine learning algorithms across metrics such as accuracy and precision in addition to utilizing a diabetes dataset.[9] The third paper, "An empirical comparison of supervised machine learning techniques in bioinformatics", also compared 10 algorithms with the following metrics: accuracy, specificity, sensitivity, and positive predictive value. One strength of this paper was providing some "Rules-of-thumb" to answer some predicted questions one might have on this paper. These answers gave us insight on how we can better prepare our paper to answer some common questions and how we can better select our algorithms.[12]

We also investigated papers that gave us a better understanding of the algorithms that we were planning on using. The fourth paper titled "A Comparative Study of Training Algorithms for Supervised Machine Learning" documents the pros and cons of five different classification techniques using metrics such as accuracy, speed, and tolerance. This paper provides elementary knowledge on each of these algorithms, while providing reasons and situations for each of these algorithms to be utilized.[1] The last paper we referenced was the "COMPARISON OF MACHINE LEARNING ALGORITHMS RANDOM FOREST, ARTIFICIAL NEURAL NETWORK AND SUPPORT VECTOR MACHINE TO MAXIMUM LIKELIHOOD FOR SUPERVISED CROP TYPE CLASSIFICATION." One strength of this paper is that they started with the "state of the art" methodologies and results from existing machine learning algorithms, giving us insight on what existing papers have demonstrated. This paper also used the same algorithms as us, including Naive Bayes, Random Forest, Neural Networks, and Support Vector Machines. For each of these algorithms, they utilized many different parameters, showcasing which ones worked best for them which strengthened their results.[8]

4 Dataset and Features

Both datasets for the project come from the UCI Machine Learning repository, and are both related to patient data who may or may not have diabetes.

The first dataset, CDC Diabetes Health Indicators[14], initially contained 21 features - 6 of which were integer types, and 15 of which were binary. Among the 253,680 entries, 13.8% of them were positive for diabetes, and 86.2% of them were negative for diabetes.

The second dataset, Diabetes 130-US Hospitals For Years 1999-2008[15], initially contained 47 features - 7 of which were integer types, 29 of which were binary types, and 38 of which were categorical string types. Among the 101,766 entries, around 14% were positive for diabetes and 86% were negative for diabetes.

We joined the two datasets in the following method: for each row of data in the first dataset, find rows in the second dataset whose age group, sex, and diabetes diagnosis matched that of the first dataset, then randomly sample one of these matching rows from the second dataset, then append it to the row from the first dataset. After appending, we get a dataset with over 250,000 entries, and with 76 features. Upon further data analysis, many of the columns were dropped - those with duplicate information, those whose values were not explained well enough to have value in our models, and those whose null count exceeded a significant threshold of more than 2% of entries. This left us with 50 attributes to use as input in training our models.

Finally, because of the imbalance in diabetes diagnosis (14% versus 86%), in selecting data for training in our models, we primarily utilize a 40%-60% split for those positive to those negative for diabetes, occasionally using a 50%-50% split for additional experimentation.

Given the large number of relevant features, most of which are binary, we experimented with Principal Component Analysis (PCA) in hope of reducing the number of features via PCA transformation in order to train some of the models faster. With PCA, we can see what features contribute the most variability in prediction, and we can select those features as our principal components, or sole features in the pursuit of binary classification. Python’s scikit learn library for PCA provides a PCA transform tool which allows us to transform our dataset from 50 features down to however many features we choose. This reduces the number of features in our training set while preserving the majority of the representation of each

datapoint. In most models, we utilized four principal components in order to more efficiently train our models.

Overall the combination of these two datasets facilitates the purpose of this project which is to analyze the pros and cons between multiple binary classification machine learning models in the context of medical practices.

5 Methods

For our purposes of replication, we chose to perform the same algorithms as the original paper, leading us to the following machine learning algorithms: Support Vector Machine (SVM), Random Forest (RF), Radial Basis Function Neural Networks (RBF NN), K-Neural Network (KNN), Logistic Regression (LR), Naive Bayes (NB), and Decision Trees (DT).

The Support Vector Machine (SVM) algorithm categorizes data points into different classes. It does this by identifying a hyperplane or line that optimally separates the points into groups. This hyperplane is established by maximizing the margin between the classes, which is the distance between the hyperplane and the nearest data points from each class. In cases where the data is not easily separable, SVMs use kernel functions to find the hyperplane. The kernel functions aid in mapping the data into higher-dimensional spaces, where class separability is enhanced, allowing for more effective classification.

The Naive Bayes (NB) algorithm classifies by assuming that features within a class are independent of each other. It utilizes Bayes' theorem, which calculates the probability of a hypothesis given the data and prior knowledge. However, this assumption of feature independence may not always hold true in real-world scenarios. Its performance can be enhanced when paired with kernel functions that estimate the probability density function (pdf) of the input data.

The K-Neural Network (KNN) algorithm utilizes distance between data points as a metric in order to determine the resulting classification (label). It starts off by choosing a number of K values and then assigning labels to the data. For every new point, the algorithm will utilize a distance formula, such as Manhattan or Minkowski, to find the nearest neighbor and then assign the label accordingly.

The Decision Tree (DT) algorithm is meant to make accurate predictions by continuously dividing the data features into classes until it reaches a point where the end criterion has been satisfied such as it is no longer beneficial to continue the division, or until it reaches a designated stopping point. Through the repetitive splitting of the data, the algorithm learns the rules for each split and as such, can predict and classify the results for new data based on the learned rules.

The Random Forest (RF) algorithm is very similar to the DT algorithm but differs in the sense that it is a collection of multiple independent trees rather than one single tree where the final prediction is formed after averaging each of these trees. Due to this design, RF is more generalizable and has less overfitting.

The Logistic Regression (LR) algorithm works to predict and classify the outcome by using a logistic curve to divide the data. Rather than simply drawing a line through the data to split it like in linear regression, logistic regression uses the curve and fits it to the data to best predict the classes of the data points.

The Radial Basis Function Neural Network (RBF NN) algorithm is a neural network architecture whose activation function is the radial basis function. The radial basis function is well-suited for approximation in non-linear, high-dimensional data models. Each neuron within an RBF NN can be associated with a particular center point in the input space, and the activation of each neuron corresponds to the similarity between the input and its associated center point (*c*). As the network passes through each iteration, parameter weight *beta* and center parameter *c* get updated via backpropagation.

$$\rho\big(\|\mathbf{x}-\mathbf{c}_i\|\big)=\exp\Big[-\beta_i\|\mathbf{x}-\mathbf{c}_i\|^2\Big]$$

6 Results

For each algorithm, we utilized a variety of hyperparameters in order to see which combination of parameters worked best for our model in regards to the accuracy, precision, and f1 score.

SVM: we explored different configurations such as linear , polynomial, and radial basis function. Within the polynomial kernel, we conducted a search to identify the optimal degree parameter.

RF: we implemented 10 max depths ranging from 1 to 10, and two criterions, gini and entropy. We then used grid search for the best parameters.

RBF NN: we experimented with 90 combinations of models, toggling between Adam and stochastic gradient descent for the optimizer (2 optimizers), a range of 3 to 10 principal components (8 different transformations for the models that utilized PCA as well as combinations that didn't utilize PCA, and 5 different learning rates (0.001, 0.05, 0.01, 0.5). By using different learning rates, we could observe what learning rates yielded better results faster, and what learning rates caused the model to either fail or fall off track. For this model, we had to use a batch size of 32 due to resource constraints. Using a larger batch number failed to produce any model as the runtime would exceed memory limitations.

KNN: we utilized odd values starting from 1 to 29 for k values, followed by four distance metrics: euclidean, manhattan, chebyshev, and minkowski. We then used grid search for the best parameters.

NB: we employed various models to predict the distribution of the data. The models used were Gaussian, Multinomial and Bernoulli.

LR: we used two solvers, lbfgs and saga, and two penalties, l1 and l2. Our c values varied with factors of 10 starting from 10^{-4} to 10^{4}. Given that not all solvers work with penalties, we then built a pipeline with a search space that contained the possible combinations of solver, penalty, and c values. From there, we did a grid search with a 10 split Stratified K fold.

DT: we implemented 10 max depths ranging from 1 to 10, and two criterions, gini and entropy. We then used grid search for the best parameters.

Table 1: Original without PCA

Algorithms	Accuracy	Precision	F1
KNN	0.8590	0.5397	0.1275
RF	0.8637	0.6792	0.1468
DT	0.8621	0.5467	0.2819
LR	0.8665	0.5979	0.2924
SVM	0.8697	0.6559	0.1486
NB	0.7221	0.2936	0.6992
RBF NN	0.7697	0.7484	0.7829

Table 2: PCA

Algorithms	Accuracy	Precision	F1
KNN	0.8607	0.4975	0.2614
RF	0.8601	0.5263	0.2660
DT	0.8589	0.5498	0.1037
LR	0.8592	0.4947	0.1250
SVM	N/A	N/A	N/A
NB	0.8199	0.3243	0.2672
RBF NN	0.6349	0.5914	0.7129

Table 3: Confusion Matrix

Algorithms	Confusion Matrix	TN, FP, FN, TP
KNN	[43061 446] [6706 523]	True Negative: 43061 False Positive: 446

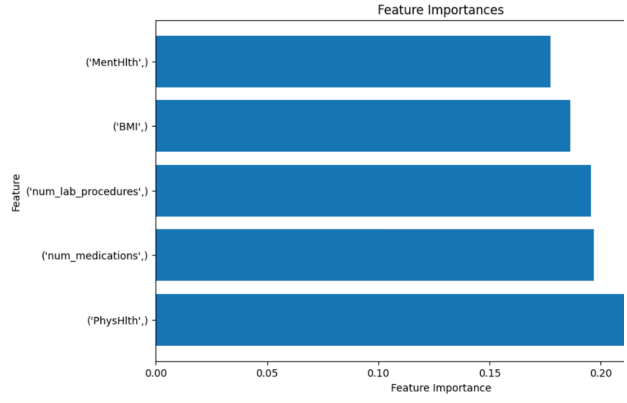
		False Negative: 6706 True Positive: 523
RF	[43226 281] [6634 595]	True Negative: 43226 False Positive: 281 False Negative: 6634 True Positive: 595
DT	[42477 1030] [5839 1390]	True Negative: 42477 False Positive: 1030 False Negative: 5839 True Positive: 1390
LR	[42566 941] [5830 1399]	True Negative: 42566 False Positive: 941 False Negative: 5830 True Positive: 1399
SVM	[53915 589] [7581 1335]	True Negative: 53915 False Positive: 589 False Negative: 7581 True Positive : 1335
NB	[51544 3113] [6920 1843]	True Negative: 51544 False Positive: 3113 False Negative: 6920 True Positive: 1843
RBF NN	[2762 775] [949 2583]	True Negative: 2762
		False Positive: 775 False Negative: 949 True Positive: 2583

For both tables, we highlighted the results from the highest performing model of each algorithm type. SVM and NB have the highest number of false negatives which suggests a struggle with correctly identifying instances where having diabetes is positive. RF and LR show a balance between the true positives and false negatives. DT shows a higher false positive count compared to true positives which indicate a challenge to correctly identify diabetes. RBF NN has the highest true positive counts indicating a better performance in correctly identifying positive cases. Similarly the RBF NN has the lowest proportion of false positives and false negatives to true positives to true negatives out of all the models, suggesting that even with unbalanced data, the model can more easily find patterns in the data for stronger classification.

When building the models on the dataset without using PCA, but using min-max scaling, SVMs yield the highest accuracy. However, all models seem to yield a very low precision and F1 score. The exception to this is RBF neural networks, whose accuracy may not be high, but whose precision and F1 score are still among the highest, along with a relatively high accuracy.

When building the models on the dataset using PCA transformation as well as min-max scaling, KNNs yield the highest accuracy. The RBF neural network does have the lowest accuracy, of 0.6349 which is lower than the neural network without PCA, but yields the highest precision and F1 score across all models even after applying PCA.

Figure 1: Top 5 Feature Importance Decision Trees



The top features were chosen by PCA and were evaluated on their importance on the DT model- shown on Figure 1. The features displayed comparable levels of importance ranging from 0.18 to 0.24. These levels show a meaningful contribution to the decision-making process of the model.

7 Conclusion

For each of our models, PCA did not yield improvements based on the metrics of accuracy, precision, and F1 score. Among the models tested, SVM exhibited the highest accuracy, while the highest precision and F1 score were observed in the RBF NN model. PCA could still be considered if prioritizing training costs over the discussed metrics is crucial. For most models, applying PCA transformation on the data reduced training time. Given that the second-highest accuracy of 76.96% was observed in the RBF NN, we can confidently consider RBF NN as our top choice for the purpose of binary classification in medicine. Further improvements can be achieved by continuing to train our best models with different datasets, and optimizing more hyperparameters.

8 Contributions

Kevin worked on the Exploratory Data Analysis to examine the features and their distributions, the data pre-processing to balance the data, and PCA for feature extractions. He then wrote the code for implementing: K-Nearest Neighbor, Decision Trees, Random Forest, Logistic Regression. He used those four algorithms on the original data and then with the PCA transformed data. From there, he began adding more descriptions to the 1st progress report risks and created the presentation slides adding details and briefs for each section. He dove into researching and documenting literature for the reports and references, along with existing papers, that were then utilized in the final report.

Marlene worked on the cleaning and preparation of the remaining two datasets to use them for training. This involves transforming the categorical values of 'yes' and 'no' into numerical counterparts, specifically 1 and 0, respectively. She also applied one-hot encoding to enhance the dataset. In parallel, she concentrated on the implementation of Support Vector Machine (SVM) and Naive Bayes algorithms. For SVM, she is experimenting with various parameters, including different learning rates, to optimize performance. Regarding Naive Bayes, she explored different models such as Multinomial and Bernoulli. She actively contributed by incorporating data into the presentation slides. She then focused on refining the data analysis. This includes exploring methods to effectively combine datasets and creating a visual representation, such as a graph, that illustrates the relationship between the cost of training and accuracy.

Aleksa worked on the merging of the datasets on similar features. For example, for the set of patients in the first dataset that are positive for diabetes, finding a way to artificially augment these data with data from the other two datasets in which the patients are also positive for diabetes (and similarly and respectively for patients that don't have diabetes). Using the data prepared by Marlene, Aleksa built multiple neural network models using the raw data (both augmented and not augmented) and the extracted data. In addition to building the neural network models, Aleksa also built the RBF decision trees, and explored other modes of data refinement for the data augmentation process.

9 Reflections

In building the models, one consideration would've been to either include a third dataset to augment onto our available data in order to have more features, or

to find a separate dataset altogether that included a richer set of attributes while still balancing subject counts for training.

Another consideration would've been to implement multi-class prediction. Since our data did originally include features such as "metformin-up" and "metformin-down", which would indicate whether a patient's prescription for diabetes had increased or decreased, we could've gone further to implement predicting what level of severity of diabetes a patient might have such as 0, 1, 2, 3 for no diabetes, low severity, middle severity, and high severity. With the hopes of generalizing these results to disease diagnosis, this consideration would've been good for going beyond just diagnosing a patient for being positive for a disease because many other illnesses may also be treated differently based on the level of severity, which could require different approaches in treatment.

Finally, we could've also looked at other evaluation metrics such as MSE, MAE, recall, ASE, etc. so that we had more substance to compare our values with and make a stronger argument for how some models are better than others for specific purposes. Additionally, we could've tried to go even further with model-building and tried other types of classification models.

10 References

[1]Bhavsar, Hetal, and Amit Ganatra. "A comparative study of training algorithms for supervised machine learning." International Journal of Soft Computing and Engineering (IJSCE) 2.4 (2012): 2231-2307.

[2]Caruana, Rich, and Alexandru Niculescu-Mizil. 2006. "An empirical comparison of supervised learning algorithms." In Proceedings of the 23rd international conference on Machine learning (ICML '06). Association for Computing Machinery, New York, NY, USA, 161–168. https://doi.org/10.1145/1143844.1143865

[3]CDC. "CDC Diabetes Health Indicators." UCI Machine Learning Repository, doi.org/10.24432/C53919. Accessed 10 Jan. 2024.

[4]Clore,John, Cios,Krzysztof, DeShazo,Jon, and Strack,Beata. (2014). Diabetes 130-US Hospitals for Years 1999-2008. UCI Machine Learning Repository. https://doi.org/10.24432/C5230J.

[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007

[7]McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

[8]Nitze, I., U. Schulthess, and H. Asche. "Comparison of machine learning algorithms random forest, artificial neural network and support vector machine to maximum likelihood for supervised crop type classification." Proceedings of the 4th GEOBIA, Rio de Janeiro, Brazil 79 (2012): 3540.

[9]Osisanwo, F. Y., et al. "Supervised machine learning algorithms: classification and comparison." International Journal of Computer Trends and Technology (IJCTT) 48.3 (2017): 128-138.

[10]Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[11]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[12]Tan, Aik Choon, and David Gilbert. "An empirical comparison of supervised machine learning techniques in bioinformatics." (2003).

[13]Uddin, Shahadat, et al. "Comparing different supervised machine learning algorithms for disease

prediction." BMC medical informatics and decision making 19.1 (2019): 1-16.

[14]Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

[15]Waskom, M. L., (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021, https://doi.org/10.21105/joss.03021.

11 Reply to Review

From the review by Group 3, they explained that some of the weaknesses of our topic was that: the feature extraction was vague, there were too many models to follow, and that no exploratory data analysis was shown. Additionally they added that we could have focused on more hypertuning rather than literary examples and more definitive feature extractions.

To revise our paper, we have added further details on how we utilized our datasets and performed feature extraction, primarily Principal Component Analysis. While there may have been confusion on the many models, it is important to show the different effects of each algorithm on the data and the results according to the metrics. To address this confusion, we have added more information on each of the algorithms so the reader can get a better understanding of each algorithm and be able to better follow along.

In their review, Group 6 suggested showing how each feature affects the model, especially in the context of our Decision Tree model. Additionally, they suggested that we provide explanations for why we chose each model.

The merged dataset has 76 features which makes it difficult to illustrate the effect of each feature on the models. However, considering their input, we have shown the top features selected by PCA and their effects on the model of Decision Trees. This is shown in Figure 1 which we added in the Results section. Although it would have been possible to show the significance of the features across our other models, we believe demonstrating their importance within one model sufficiently establishes their significance in the decision-making process.

In our methods section, we outlined the benefits of each model to clarify our decision-making process. Our main aim was to compare different models and find the one with the best metrics.