Can we identify whether a patient has diabetes or is healthy from the UCI CDC Diabetes Health Indicators dataset?

Pulindu Fonseka, *cfcf66, 18/01/2024*

Abstract—I used the UCI CDC Diabetes Health Indicators dataset to predict whether a patient is diabetic or healthy using categorical lab test results and answers to survey questions. The dataset is a very useful resource, and, in this paper, I will explain how I used it and made a working AI model to predict whether a patient is diabetic. I'm very passionate about this topic because I personally know many people with diabetes and if some of them are notified of their health status earlier, they may have a better quality of life with the help of modern treatments.

Index Terms—K-Nearest neighbours, Linear SVC, One-Hot Encoding, SMOTE

1 Introduction

FOR For this machine learning assignment, I have decided to use a dataset from the UCI datasets, the CDC Diabetes Health Indicators dataset to be exact. The machine learning task that I am going to complete is: 'Can we identify whether a patient has diabetes or is healthy from the UCI CDC Diabetes Health Indicators dataset?'. I'm going to be using the dataset to predict whether 'Diabetes_binary' is 0 or 1. This is a binary datatype, and I will use classification methods to answer the machine learning question.

There are many health conditions associated with diabetes, including coeliac disease, thyroid disease, haemochromatosis, heart disease and dental problems. My motivation for choosing this question is that if the models created are good enough, many people worldwide could use a model like mine in the future to quickly and cheaply determine whether they have diabetes, and a similar model can be used to assist doctors and nurses worldwide to easily diagnose patients of a range of diseases and health conditions. Thus, lessening the workload of doctors and nurses worldwide. We are a long way away from that point, but I hope that my work makes a step forward in the right direction.

2 Dataset Analysis

The CDC Diabetes Health Indicators dataset contains 22 features and 253,680 instances. The features are mainly multiple-choice questions for the patient to answer about their lifestyle, but it does also contain official lab test results. All the features are either categorical or binary.

After analysing the dataset, I can see that there are no null values in the dataset, nor are there outliers making data cleaning and pre-processing simple. Since the data has clearly already been pre-cleaned. The patient ID has also been removed, so, individual patients can't be uniquely identified. This is useful because it avoids risking breaching people's privacy. However, there is a potential issue with the dataset: the 'Diabetes_binary' column is very skewed. Only 13.9% of the patients in the dataset are reported to have diabetes. Furthermore, the patients that are classed as pre-diabetic will be classed as not having diabetes. These factors will affect my models, but there are steps that can be taken in advance to improve the model.

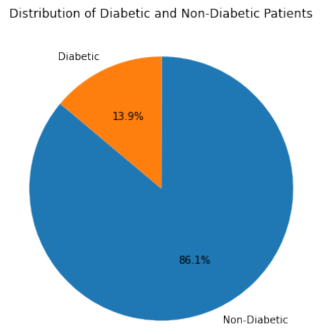


Fig. 1. Pie chart to show proportion of diabetic/healthy patients

3 Pre-Processing and Set-Up

The first step I made was to encode the categorical data to ensure that they hall in the range 0-1. I used one-hot encoding. The reasons I used this encoding are that firstly it works well even when there are a lot of categories (so that it can be easily modified to allow datasets when there are a lot more categories). It's simple to implement and use while preserving the information correctly. Also, I transformed the dataset and converted all floats to integers. Since it's categorical it won't make a difference to functionality, but the data is more readable as integers, some machine learning models work better when categorical data has been one-hot encoded.

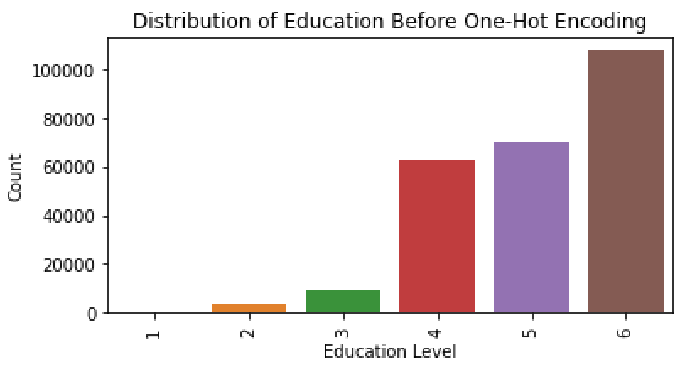


Fig. 2. Distribution of Education column before One-Hot encoding

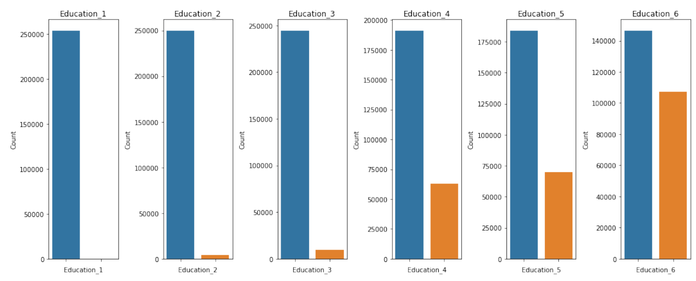


Fig. 3. Distribution of Education column before One-Hot encoding

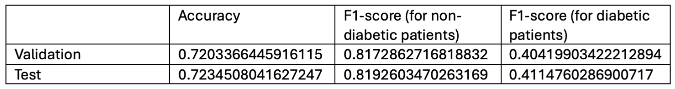
Following that, I split the dataset into a test set and a training set. The split I have chosen is a 0.56 : 0.14 : 0.3 split of training set, validation set and test set, respectively. I did this because I wanted a large enough test to set to ensure the unbiased evaluation of the data, and since only 14% of the patients are diabetic, I wanted a large test set, so I chose 30%. For the remaining 70%, I wanted an 80% training set and a 20% validation set. It seemed like a sensible split to me.

4 Model Evaluation

The machine learning models I used were from the scikitlearn library. And for both models I used a manual grid search to optimise my hyper-parameters. I'm going to prioritise an improvement in F1-score for diabetic patients because for the task I have chosen, it is better to have false positives than false negatives, since in the worst case it'll just lead to people asking for further medical attention. But false negatives could possibly be dangerous in the medical field. The 2 machine learning models I have chosen for this task are SVC and K-Nearest-Neighbours. Both are popular for large data sets.

4.1 Linear SVC

The reason I chose Linear SVC is because the dataset I have chosen is very large, and SVC works very well in high-dimensional spaces. However, it's sensitive to unbalanced data. Since my dataset is very unbalanced (only 13.9% of patients are diabetic), I decided to implement SMOTE. SMOTE generates synthetic samples from the minority class to make the dataset more 'balanced', vastly improving performance of the model. SMOTE was implemented on the training data and then I ran a for-loop to find an optimal C value (a regularisation parameter) without over-fitting nor under-fitting. The for-loop then tests the model against the validation set. The data is printed in each iteration, and I created a CSV file with it for analysis. At C=0.001, it's best at predicting when a patient is diabetic, however it's drastically over-fitting and so it's not an optimal model. At C=0.00483293023857175, it's got the next highest F1-score when the patient is diabetic, so I decided to use this as my optimal hyper-parameter for SVC. When I used the model against my test set, I get the following results:



	Accuracy	F1-score (for non-diabetic patients)	F1-score (for diabetic patients)
Validation	0.7203366445916115	0.8172862716818832	0.40419903422212894
Test	0.7234508041627247	0.8192603470263169	0.4114760286900717

Fig. 4. Table to show results

Furthermore, since I used one-hot encoding, my linear SVC model will have a higher accuracy than if I didn't use the encoding. This is because linear SVC is sensitive to scaling of input values, so, encoding the data reduces the negative impact.

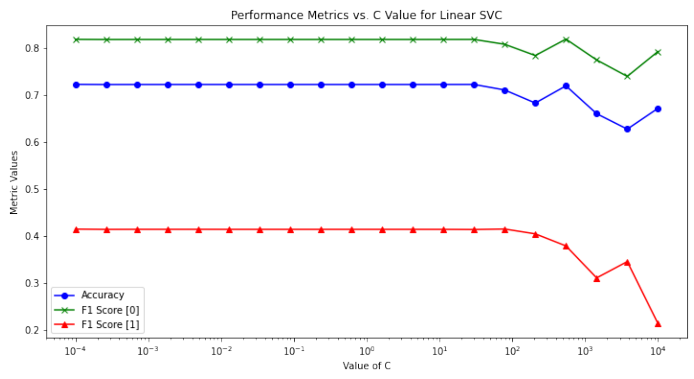


Fig. 5. Performance of SVC

4.2 K-Nearest-Neighbours

I picked KNN because it's a simple and easy to interoperate model. The hyper-parameter I chose to tune is the value k. To get an optimal k-value for my model, I used a manual grid search. Put simply, it's a for-loop that tries every value of k in a range (mine was from 1-60) and prints the results. I analysed the results to find the most optimal k-value. When k=1, the f1-score for diabetic patients is 0.3, which is the highest in the range, however it wasn't a suitable k-value to choose because it's over-fitting the data since it's only considering 1 close value. Therefore, I chose a k=5, since the f1-score is 0.27. For values larger than 5, the model gets progressively worse, (as shown in the graph).

For larger values of, they are over-fitting the model. The model eventually predicts for all patients to be non-diabetic since that's the majority class. I believe this could be reduced if I used SMOTE with KNN too since then the effect of an unbalanced dataset will be reduced.

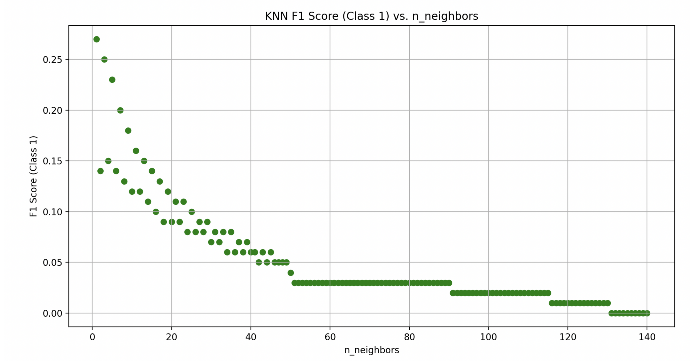
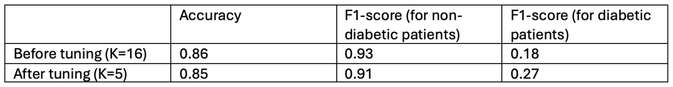


Fig. 6. Performance of KNN



	Accuracy	F1-score (for non-diabetic patients)	F1-score (for diabetic patients)
Before tuning (K=16)	0.86	0.93	0.18
After tuning (K=5)	0.85	0.91	0.27

Fig. 7. Table to show results

After tuning, the accuracy and F- score for non-diabetic patients has gone down, but the F1-score for diabetic patients has increased significantly (up by 9%). A limitation on using KNN is the large computational time it takes for such a large dataset, making fine-tuning hyper-parameters more time-consuming.

5 Conclusion

The machine learning task that I completed is ‘Can we identify whether a patient has diabetes or is healthy from the UCI CDC Diabetes Health Indicators dataset?’. My best model has an accuracy of 72.3

My model suggests that there is a correlation between the categories (such as BMI, high cholesterol, and heart disease/attacks) and risk of diabetes, and the medical literature does support the conclusion. However, my models don’t perform as well as I had hoped. The better of my 2 models is SVP, I believe this is the case for 2 reasons. Firstly, because SVP is specifically used by many people because it’s suitable with very large datasets. Secondly (as I mentioned in section 4) I didn’t use SMOTE or any similar methods for KNN.

As a future improvement I’d spend longer pre-processing my data. More specifically, I’d use SMOTE on KNN. Another change I’d make is removing less-related features, such as mental health. Mental health may have a small correlation with diabetes, but for my models I believe that columns that are less related to diabetes and physical health has negatively impacted my models since they may be overly sensitive to it. Furthermore, my models may perform better with a dataset that has extra features that are scientifically known to be a factor for diabetes. For example, according to Web MD, certain races are more likely to develop type 2 diabetes than others, so including extra columns like race in the data could make the models perform better. But including race could also be more problematic as the machine learning models could disproportionally report people of certain ethnic groups as diabetic due to the skewed data. I’d find it interesting to experiment with different datasets or in the future improving on my pre-processing.

5.1 Self-Evaluation

Over the course of this module, I have learnt the basics of machine learning, and its core principles. From the coursework I learnt how important it is to rigorously pre-process the data, and that it’s arguably the most important part of developing AI models. If I was to do this coursework again, I would experiment with different models to see how much of an effect that has. I’d also implement more data preparation, such as using SMOTE, or similar, for KNN. Difficulties for the module for me include hyper-parameter fine-tuning and the time taken to do it. Finally, the unique contributions I made during this coursework include using the data set to predict a real-world problem.

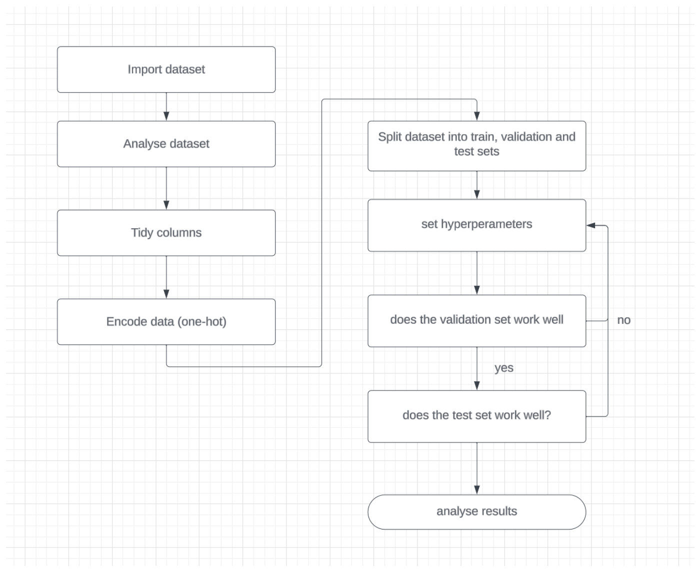

Import dataset
Analyse dataset
Tidy columns
Encode data (one-hot)
Split dataset into train, validation and test sets
set hyperperameters
does the validation set work well
no
yes
does the test set work well?
analyse results

Fig. 8. Machine Learning workflow flowchart

References

[1] ”UCI Machine Learning Repository.” Archive.ics.uci.edu, archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators.
[2] ”Diabetes Related Conditions.” Diabetes UK, www.diabetes.org.uk/diabetes-the-basics/related-conditions.
[3] McQueen, Janie. “Type 2 Diabetes: How Does Race Make a Difference?” WebMD, 21 June 2021, www.webmd.com/diabetes/type-two-diabetes-race.