Multi-modal classification of diabetic retinopathy patients using machine learning

Charles O'Neill
College of Science
Australian National University
Canberra, ACT 2601
charlie@macuject.com

October 2, 2021

Abstract

Diabetic retinopathy (DME) is orders of magnitude more complex than neovascular macular degeneration in terms of treatment complexity. It is difficult to determine what factors contribute to achievement of long-term clinical outcomes for patients. Similarly, erratic treatment protocols make it challenging to classify and correct doctor decisions. Here, we apply both supervised and unsupervised learning techniques to (1) assess the characteristics of treatment that lead to good patient vision; (2) determine how accurately we can predict visual changes with minimal data; and (3) group doctor protocols in a definitional way. This research may be used as a springboard for clinical decision-support tools.

1 Introduction

The goal for tasks (1) and (2) above was to classify a patient's most recent vision as being either above or below baseline vision. *Overall visual change* (OVC) is defined as the signed difference between a patient's most recent vision, and their visual acuity at the initial appointment. Therefore, the task became predicting whether OVC for any given patient would be positive or negative at some chosen future time.

Task (3) is about providing a proof-of-concept for dimensionally-reducing complicated interval combinations in order to visualise doctor protocols.

2 Materials and method

We are provided with three datasets from three distinct doctors. The datasets were concatenated into one, with an additional variable added that identified which clinic the patient attended. For two of the doctors, the clinic was shared. Each patient's longitudinal journey was condensed using an automated Python script to a single row of descriptive features, which differed depending on which task was required (1 or 2). For task 3, the only data fed to the t-SNE model were the interval combinations.

Constructing training and validation sets. Model performance was assessed using k-fold cross validation with five splits. Cross-validation is useful when there is limited data, and we wish to obtain a less biased estimate of performance on unseen data than may be portrayed by a simple train-test split. To use k-fold cross validation, the dataset was randomly divided into k groups, where $k = 5$ in this instance, of approximately equal size. For any given training run, the model is fit on the first $k - 1$ folds, and the remaining fold is used as a validation set to determine model accuracy on unseen data. Training the model on each combination of folds, we can then find the mean of the accuracy on each validation set, and also the variance of predictions.

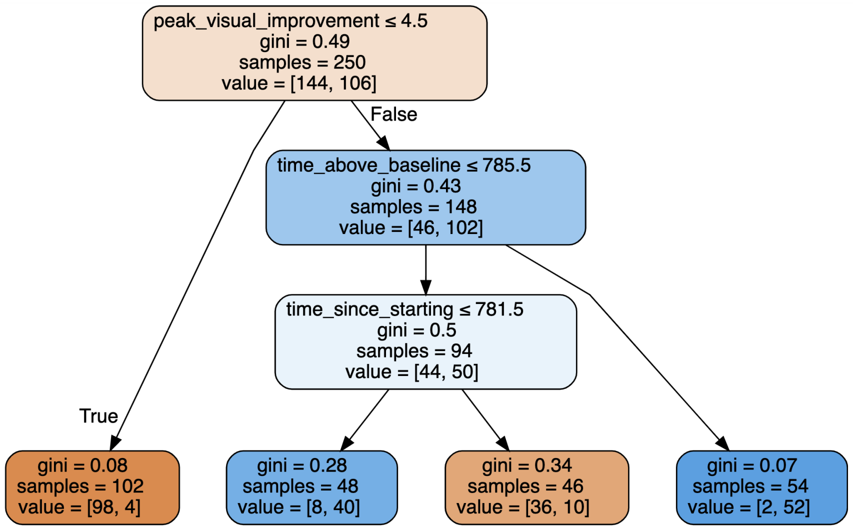

Figure 1: A simple decision tree with four leaf nodes to demonstrate feature importance.

2.1 Different features

A stated goal of this report is not just predictability, but explainability and transparency in model construction. For this reason, two different sets of features were constructed for Tasks 1 and 2 respectively:

1. A set of features that included all statistics of patient vision, minus ones where OVC could be directly inferred. These included mean vision, time above baseline, peak visual improvement, time to peak vision, baseline vision, number of visits thus far, and time since starting. The purpose of this set of features was to see which elements of a patient's whole longitudinal journey influenced overall visual change the most.

2. A different set of features that only included information from the first n visits. These included mean vision, standard deviation of vision, peak visual improvement, baseline vision, location, and initiation drug (all from only the first n visits). The purpose of this set of features was to determine how accurate a prediction of OVC could be based on the first year of treatment.

3 First feature set (entire patient journey)

3.1 Decision tree

A decision tree with four leaf nodes was formed to examine the initial accuracy of a baseline model, as well as to determine features important to prediction. This model achieved a classification accuracy of 79% on the validation set.

The decision tree is shown in Figure 1. Clearly, the most important features (at least those that can be recognised as such by a greedy algorithm such as a decision tree) are peak visual improvement (PVI), time above baseline, and time since starting treatment. Whilst the first two are directly expected, the third is more interesting; patients who have been treated for longer are likely to have worse vision.

3.1.1 Location (Box Hill, Boronia): encoding categorical variables

Normally, categorical variables are dealt with by one-hot encoding them and feeding them to an embedding layer. The embedding layer helps the model to discover the meaning of the different levels of these variables [1]. In a decision tree, we don't have embedding layers, so instead we use one-hot encoding to replace a single categorical variable with multiple one-hot-encoded columns, where each column represents a possible level of the variable. Whilst this is not recommended when there are a large number of levels, computational complexity is minimised here as there are only two locations - Box Hill and Boronia.

3.2 Random forest

In order to better interpret our models, we next construct a random forest. Random forests are based on the notion that, whilst training different models on subsets of data will lead to more inaccurate models, taking the average of all these different model predictions will cause the uncorrelated errors from different models to cancel out. This procedure is known as bagging. This means that we can improve the accuracy of nearly any kind of machine learning algorithm by training it multiple times, each time on a different random subset of the data, and averaging its predictions [2].

Our initial random forest with no hyperparameter tuning achieved 83% accuracy on the validation set, already a marked improvement over the decision tree. The random forest allows us to answer the following questions:

- How confident are we in our predictions using a particular row of data?
- For predicting with a particular row of data, what were the most important factors, and how did they influence that prediction?
- Which columns are the strongest predictors, which can we ignore?
- Which columns are effectively redundant with each other, for purposes of prediction?
- How do predictions vary, as we vary these columns?

3.2.1 Tree variance for prediction confidence

We saw how the model averages the individual tree's predictions to get an overall prediction—that is, an estimate of the value. To measure the confidence of this estimate, we could use the standard deviation of predictions across the trees, instead of just the mean. This tells us the relative confidence of predictions. In general, we would want to be more cautious of using the results for rows where trees give very different results (higher standard deviations), compared to cases where they are more consistent (lower standard deviations).

Obtaining predictions for every tree and every patient (40 trees and 71 patients) in the validation set, we find that the mean standard deviation over all trees and all patients is 0.25. Looking closely at the individual standard deviations for each patient, we can see that the confidence in the predictions varies widely. For some patients, there is no standard deviation, because all the trees agree. For others it's higher, as the trees don't agree. This information would be useful in a clinical-support setting; for instance, if tree variance was high for a particular patient, you would be wary of using the overall prediction in any clinical decision.

The standard deviations of the first five patients are shown below.

```
array([0.35707142, 0., 0.35707142, 0., 0.41758233, 0.15612495])
```

3.2.2 Feature importance

We also wish to know what the most important factors are for prediction. We can do this by examining feature importance. To calculate feature importance, we loop through each predicting tree, and at each branch recursively examine what feature was used for that split. We can then determine how much the model improved as a result of that split. The improvement is weighted by the number of rows in that group and added to the importance score for the feature chosen in the split. Finally, the scores are normalised so they add to 1.

The feature importances are shown in Figure 2. The features chosen for the splits in our initial decision tree are still the most important here. Of note, the location of the clinic in which the patient was treated is of essentially no use in constructing the random forest model.

3.2.3 Partial dependence

Now that we know the most important features, we'd like to understand how these affect predictions using partial dependence plots. These attempt to answer the question: if a row varied on nothing other than the feature in question, how would it impact OVC? That is, we wish to see how a certain feature, such as time since starting, impacts OVC, all other things being equal.

To answer this question, we can't just take the average time since starting treatment for possible value. The problem with that approach is that many other things vary from journey length to length as well, such as time to peak, mean vision, and baseline vision. Thus, merely averaging over all the patients that have the same time since starting would also capture the effect of how every other field also changed along with time since starting and how that overall change affected OVC.

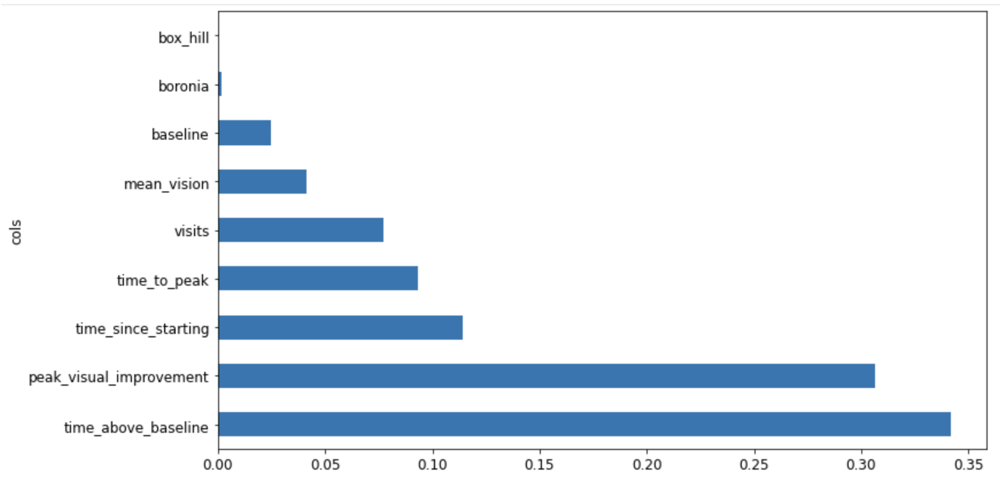

Figure 2: Feature importances for model predictors.

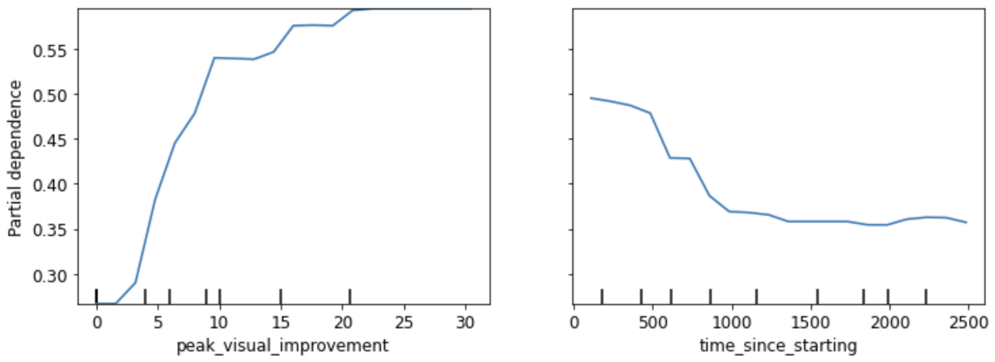

Figure 3: Partial dependence plot for two important feature predictors.

Instead, what we do is replace every single value in the time since starting column with with the minimum in the dataset, and then calculate the predicted OVC for every patient, and take the average over all patients. Then we do the same for for all days since starting treatment until the maximum time since starting. This isolates the effect of time since starting [3].

Partial dependence plots are shown in Figure 3 for two predictors.

3.2.4 Redundant predictors

In Figure 4, the pairs of columns that are the most similar are the ones merged together early on the left. Of note, the pair of predictors merged first were the one-hot encodings for Box Hill and Boronia. This tells us that the locations predict essentially the same thing. Here, similarity is determined by rank correlation.

3.2.5 Influence of features on prediction

We have already computed feature importances across the entire random forest. We now undertake a similar process on a single row of data: an individual patient. From this one patient, we input their data into the first decision tree, and note what split is used at each point throughout the tree. For each split, we see what the increase or decrease in the addition is, compared to the parent node of the tree. We do this for every tree, and add up the total change in importance by split variable.

The clearest way to display the contributions is with a waterfall plot. This shows how the positive and negative contributions from all the independent variables sum up to create the final prediction (the *net* column).

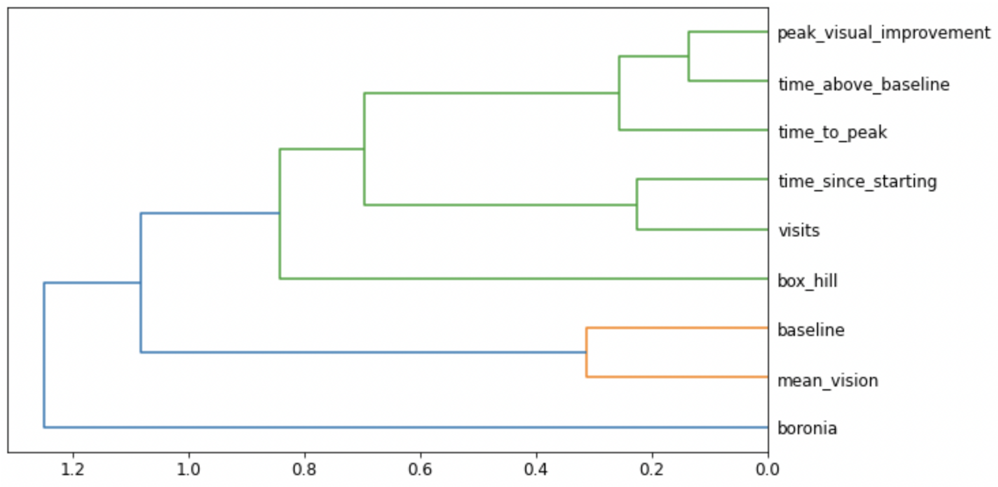

Figure 4: Redundant predictors, based on rank correlation.

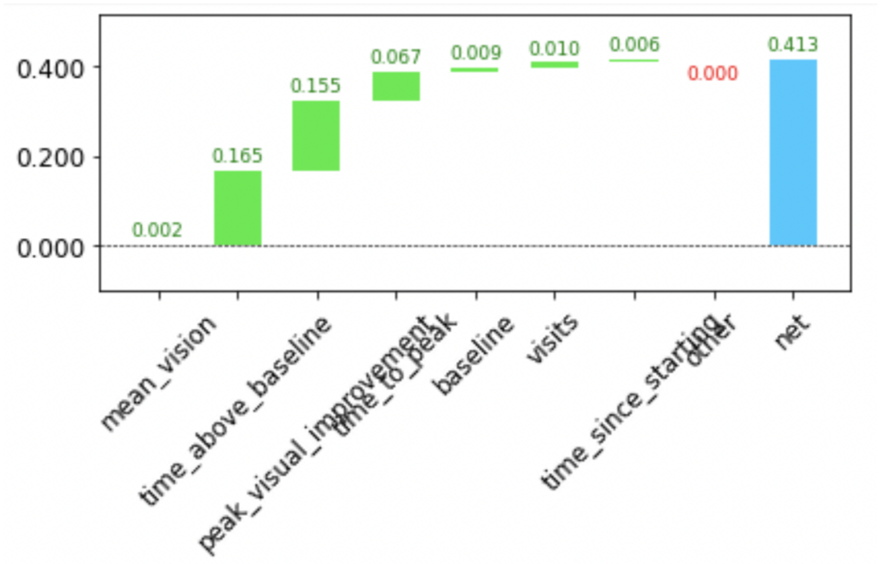

Figure 5: A waterfall plot visualising contributions of predictors for an individual patient.

Whilst slightly superfluous during research and exploratory analysis, this information would be useful in a clinic-support setting. It can be used to provide useful information to doctors and patients about the underlying reasoning behind the model's predictions.

3.3 Neural network

Training a neural network with two hidden layers (500 and 250 activations, respectively), we achieve a validation set accuracy of 89% after three epochs of training. This is likely the best performance we can achieve on this dataset, considering its small size and large inherent variability.

3.4 Summary

A summary of model results is shown in Table 1. Clearly, the neural network performs best. However, it only provides a small increase in accuracy, suggesting that there is no deep underlying relationship between variables that cannot be predicted by classical machine learning models.

Table 1: Model accuracy results (Task 1, whole journey).

Entire patient journey statistics.		
Model	Training accuracy	Validation accuracy
Decision tree	90%	79%
Random forest	96%	86%
Neural network	97%	89%

Table 2: Predictive model capacity for varying inputs and hyperparameter requirements (Task 2).

Varying number of input visits.			
No. input visits	No. required visits	Year cut-off for prediction	Validation accuracy
3	6	2	69%
4	6	2	78%
5	6	2	86%
Varying number of required visits.			
No. input visits	No. required visits	Year cut-off for prediction	Validation accuracy
4	7	2	72%
4	8	2	77%
Varying cutoff for prediction.			
No. input visits	No. required visits	Year cut-off for prediction	Validation accuracy
4	6	1	84%
4	6	2	75%
4	6	3	85%

4 Second feature set (first n visits of treatment)

As outlined above, the second goal for this exploratory analysis was to use as little data as possible to get the most accurate predictions concerning OVC, as far into the future as feasible. That is, we wish to feed our model a certain number of visits from initiation; require that the patient has at least a certain number of years of data, and a certain number of visits in those years; and finally, predict whether OVC will be positive or negative at a standardised cutoff for all patients. In doing so, we essentially have three hyperparameters to tune:

1. The number of initiation visits worth of data we feed our model.
2. The number of years of data we require the patient to have. This also coincides with the standardised cutoff for the time we are trying to predict OVC for (for instance, a two year requirement for data means we will try and predict OVC at the two year mark).
3. The number of visits we require a patient to have within the chosen cutoff. This requirement is for two reasons. Firstly, it prevents unrealistically strong model performance performance due to patients who only have one or two visits after their initiation training data, in which case predicting OVC is a fairly simple task of extrapolating from the last visit. Secondly, it ensures that we actually have enough data in the first place to feed to our model.

Thus, it makes sense to do a recursive search over these hyperparameters. We first choose a set of reasonable initiation visits, and a set of yearly cutoffs, as well as a required number of visits. We can then perform a random search with Gaussian sampling over this space, and record our results in the table below. Each model was constructed with a baseline random forest with no hyperparameter tuning, for standardisation of comparison. This is shown in Table 2.

These results are promising, and seem to suggest that high accuracy, with little input, is possible. An interesting aspect of these results is that accuracy seems to dip when we're trying to predict OVC at the end of the second year. Whilst it makes sense that accuracy for the end of the first year would evidently be higher (as there is less time for OVC to change dramatically from the initiation visits), the fact that accuracy is highest when predicting the third year of vision is surprising. This suggests that patients reach a stable and predictable level of vision that is largely determined by the first few visits of data.

Table 3: Classification metrics for predictive model.

Metric	Value
Accuracy	0.87
Precision	0.81
Recall	0.89
F1 Score	0.85

4.1 Common classification metrics for model

An important aspect of evaluating a binary classification model such as the one above is looking deeper than the simple metric of accuracy. Here, we define and calculate a variety of common classification metrics, and contextualise the model's performance. We denote the number of true positives as TP, the number of false positives as FP; and similarly the number of true negatives at TN, and the number of false negatives as FN.

The first metric is *precision*, defined as

$$\text{Precision} = \frac{TP}{TP + FP}$$

Since the denominator is equivalent to the total cases where a positive OVC was predicted, we can view precision as a representation of how accurate our predictions are for patients with positive OVC. That is, out of all the patients we predicted positive, how many actually had a positive OVC? Precision is a useful measure when the cost of a false positive is high, such as email spam.

The second metric, which often accompanies precision, is *recall*, defined as

$$\text{Recall} = \frac{TP}{TP + FN}$$

Thus, recall calculates how many of the patients with a positive OVC our model correctly finds. Recall is a useful metric when false negatives carry a high penalty, such as fraud detection.

F1 score is a metric that combines both precision and recall. It is defined as the harmonic mean of the two, representing a weighted average.

$$\text{F1} = \frac{2PR}{P + R}$$

Expressing this in terms of true positives, false positives, and false negatives, we get

$$\text{F1} = \frac{2TP}{2TP + FP + FN}$$

Calculating these metrics for our model, we get the values shown in Table 3. Our recall is higher than our precision. The model correctly identified 89% of actual positive OVC patients (recall), whilst out of the patients it predicted were positive, only 81% actually were (precision). Thus, our model is better at finding all the positive OVC patients, at the cost of mis-classifying some negative OVC patients as positive. That is, the false negative rate is lower than the false positive rate.

5 Unsupervised learning to cluster doctor protocols

We want to classify the doctors to see if there is any difference in interval patterns.

Our next step is to generate our values, which in this case are the interval length time series, one for each patient. We will extract only 10 visits, for only those patients who have actually had 10 visits.

We use t-distributed stochastic neighbour embedding to visualise our high dimensional data, introduced by van der Maaten and Hinton in 2008 [4]. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. That is, we take a set of points in a high-dimensional space, in this case the interval lengths for the first ten visits, and find a faithful representation of those points in a lower-dimensional space, typically the 2D plane. The algorithm is non-linear and adapts to the underlying data, performing different transformations on different regions.

The results are shown in Figure 6. It is clear that whilst Doctor 0 and Doctor 1 are similarly mapped in the vector space, Doctor 2 is more concentrated in the upper right-hand quadrant of the mapping. This suggests that Doctor 2 uses similar protocols for a majority of his patients; these protocols are a subset of Dr 1 and 0's protocols.

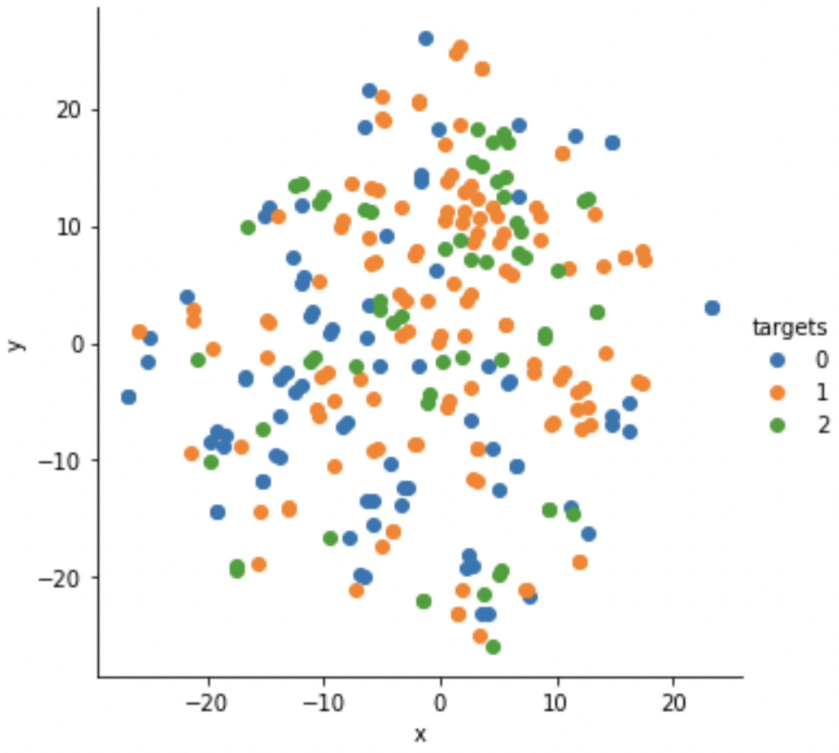

Figure 6: Using t-SNE to cluster doctor protocols.

6 Further steps

The question that needs the most fleshing out is clustering doctor protocols. The next step will be to experiment with different hyperparameters for the t-SNE representations. This will allow us to start compiling interval statistics and see what doctors are actually doing, with an idea in mind that some doctors will be more similar to others.

Another further research question is applying the same explainability tools to the predictive model, to see which aspects of the first few initiation visits are most influential on OVC later in the patient journey.

Finally, an additional feature could be added for the data in Tasks 1 and 2 which records not only clinic location, but also treating doctor.

References

[1] Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

[2] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[3] Jeremy Howard and Sylvain Gugger. Fastai: a layered api for deep learning. *Information*, 11(2):108, 2020.

[4] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.