Using Machine Learning to Predict the Onset of Diabetes
Ben Crabtree

Aim

The aim of this study is to determine a machine learning algorithm that will correctly predict the onset of diabetes in new patients with a high-level accuracy after having been trained with the Pima Indians Diabetes dataset. In order to accomplish this, we train several machine learning algorithms using the Pima dataset and determine their accuracy using stratified 10-fold cross validation. We then compare the results.

This study is important because having a tool that can predict the onset of diabetes in patients with a high degree of accuracy could be very beneficial in assisting doctors in deciding what course of treatment to recommend, particularly in ambiguous cases. Making such predictions at an early stage could save lives as treatment could begin earlier, perhaps preventing the onset of the condition or enabling patients to better manage symptoms.

Data

In this study, we use the Pima Indian Diabetes dataset to train the prediction algorithms. The dataset contains 768 observations, each of which is comprised of 8 numeric predictor variables and one nominal response variable.

Each predictor variable measures a biometric attribute gathered from medical tests, or a personal characteristic of an individual patient. These are:

1. num_pregnancies: Number of times pregnant
2. plasma_glucose: Plasma glucose concentration (derived from an oral glucose tolerance test)
3. diastolic_bp: Diastolic blood pressure (mm Hg)
4. skinfold_thickness: Triceps skin fold thickness (mm)
5. serum_insulin: 2-Hour serum insulin (mu U/ml)
6. bmi: Body mass index (weight in kg/height (in m) squared)
7. diabetes_pedigree: Diabetes pedigree function
8. age: Age in years

The values of the response variable, class, have been altered from the original dataset to be nominal:

1. yes: Tested positive for diabetes (268 instances)
2. no: Tested negative for diabetes (500 instances)

All patients from which this data was gathered are females who were at least 21 years old at the time of collection and who are of Pima Indian heritage.

In the original dataset, there are some missing values which in our training data have been filled using the mean value of the attribute to which they belong. We have also normalised the numerical values in our training data by attribute to fall within the interval [0,1] using

the Weka software. This ensures that all attributes are treated with equal importance in the training process.

We also retrained the algorithms using a version of the training dataset in which the number of attributes has been reduced using correlation-based feature selection (CFS) and remeasured their accuracy. CFS selects a subset of the original predictor variables which are highly correlated with the response variable but not with each other. After running CFS on the training data in Weka, the predictor variables that were chosen were: plasma_glucose, serum_insulin, bmi, diabetes_pedigree and age. This seems reasonable as a strong set of predictors for diabetes, but it doesn't seem likely that they are completely uncorrelated with each other.

Results and discussion

We compare the accuracy of 8 different algorithms:

1. Zero Rule (ZeroR)
2. One Rule (1R)
3. k-Nearest Neighbour (1NN and 5NN)
4. Naïve Bayes (NB)
5. Decision Tree (DT)
6. Multi-Layer Perceptron (MLP)
7. Support Vector Machine (SVM)
8. Random Forest (RF)

The first table records the accuracy of Weka's implementations of these algorithms. We measure the accuracy of each using stratified 10-fold cross validation on both the full Pima training set and the streamlined CFS training set.

	ZeroR	1R	1NN	5NN	NB	DT	MLP	SVM	RF
No feature selection	65.10	70.83	67.84	74.48	75.13	71.74	75.39	76.30	74.87
CFS	65.10	70.83	69.01	74.48	76.30	73.31	75.78	76.69	75.91

ZeroR and 1R are baseline algorithms against which we compare the accuracy of the other algorithms. For the full dataset, we can see that ZeroR and 1R are the least accurate, as expected, meaning that these simple algorithms are underfitting and not capturing the underlying patterns in the data accurately enough. However, they still perform much better than a random guess. 1R performs better than ZeroR as 1R has slightly higher capacity. ZeroR simply counts which class appears more in the training data and assigns any new test data to this class. 1R relies on a majority class vote using the single attribute for which this vote leads to the fewest misclassifications in the training set. Given that neither of these methods rely on multiple attributes, it's not surprising that CFS does not improve their accuracy at all.

For KNN we look at accuracy using two different values of k. In 1NN, k=1 and in 5NN, k=5. 1NN does not perform as well as 5NN, which is to be expected as 1NN will tend to overfit to

the training data, as it only considers the single training point closest to a new example, which leads to higher variance and negatively impacts accuracy on the test data. 5NN reduces this variance somewhat by looking at the 5 nearest neighbours of any new example. CFS improves the accuracy of 1NN somewhat but doesn't improve 5NN at all. This makes sense as CFS results in a dimensionality reduction, from 8 to 5, which can often improve accuracy for methods that rely on notions of distance, as KNN does. Eliminating attributes lowers the number of dimensions and alters which training points are 'nearest' to any new example, likely making them more similar, meaning the single closest neighbour of a new example will be more likely to share a class with that example. 5NN is less affected by this strategy as variance is already mitigated by the majority vote of the five nearest neighbours.

Naïve Bayes performs better than 1NN and 5NN and a single decision tree, which is to be expected as decision trees on their own do not tend to have particularly high accuracy. CFS improves accuracy of Naïve Bayes, which makes sense. NB asks – what is the probability of a new example being in a certain class given the evidence (predictor values for new example)? If the evidence is more streamlined, with only the 'best' (ie most correlated with class, least correlated with each other) predictor variables included as in CFS, then the accuracy of the NB prediction will increase. CFS also improves the accuracy of a single decision tree, again by using attributes which are not correlated with each other.

Not surprisingly, the random forest has higher accuracy than a single DT due to the variance smoothing effects of using an ensemble of decorrelated decision trees. CFS slightly improves accuracy of the RF further, creating decorrelated trees by training each on a random subset of attributes which have already been selected for their low correlation with each other. In this way, the accuracy of each tree is improved, and the majority vote of these trees will have lower variance and therefore higher accuracy.

The multi-layer perceptron performs slightly better than Naïve Bayes and Random Forest. Weka uses a sigmoid function by default as the activation function. This non-linear activation in combination with several hidden nodes means the decision boundary can be non-linear and quite flexible, so it is not surprising that this bump in model capacity has resulted in slightly higher accuracy. The improvement to performance with CFS is negligible. The CFS training data did result in 4 hidden sigmoid nodes being used in the network – fewer than the architecture generated for the full training set, which used 6. This suggests that a better training set means a more efficient architecture can be used with slightly more accurate results.

The Support Vector Machine is the best performer out of all the algorithms. The default kernel selected by Weka is linear and the good accuracy with this kernel suggests that there is no need to transform the data to a higher dimension in order to make it linearly separable. If we run SVM with an RBF kernel for example, we get a much lower accuracy of 65.10% for both the full dataset and CFS selected attributes. A lower number of variables after CFS means that the data exists in a slightly lower dimensional space but this does not seem to have greatly impacted the results with the linear kernel as CFS improves SVM's accuracy by a small amount, suggesting that the data was linearly separable in the 5 dimensions described by the 5 variables selected by CFS, and continued to be linearly separable in the 8 dimensions described by the full set of variables.

The second table contains accuracy results for implementations of 1NN, 5NN and Naïve Bayes in python, with accuracy measured using stratified 10-fold cross validation, also implemented in python.

	My1NN	My5NN	MyNB
No feature selection	68.49	74.47	74.74
CFS	69.15	75.39	76.18

The python implementation of 1NN is slightly more accurate than the one in Weka, and CFS improves that accuracy further. The accuracy of python 5NN is comparable to Weka's implementation, and with CFS the accuracy exceeds Weka's with CFS. Euclidian distance is used as a distance function in both the python and Weka implementations, so this increase in accuracy of the python implementation isn't accounted for by a difference there. Perhaps it is due to the level of precision to which the distances are calculated and compared. The python implementation of Naïve Bayes is not as accurate as Weka's without CFS, although with CFS the accuracy becomes comparable to Weka's accuracy with CFS. This could be due to a greater degree of precision when calculating probabilities in Weka's internal implementation of NB.

We see that overall, CFS was beneficial to the accuracy of most of the classifiers, and it was never harmful. In only some cases did it not produce an increase in accuracy. There is an argument that since this method selects only some predictor variables on which to train the algorithms, we are basically discarding data, but the counter argument is that this data may be more noise than useful signal. If certain variables are not highly correlated with the response, they may not be beneficial to include. If they are highly correlated with other predictor variables, they may be redundant. Using CFS also has the added benefit of reducing the dimensionality of the data – in this case from 8 to 5 – aiding in mitigating the curse of dimensionality, which can be a problem in high dimensional data, especially for algorithms relying on notions of distance or fitting decision boundaries in higher dimensional space.

Conclusion

In conclusion, the algorithm with empirically greatest accuracy on the Pima dataset, as measured by stratified 10-fold cross validation, is the Support Vector Machine. We found that using correlation-based feature selection improved the accuracy of most algorithms further, although the improvement to SVM was small. SVM was closely followed by Weka's Naïve Bayes implementation, and the python NB implementation was very close behind that. If choosing a prediction algorithm purely on accuracy, SVM trained using CFS filtered data seems like the best choice.

However, pure accuracy might not be the only consideration necessary in this application. As we seek to make predictions in the medical domain, interpretability could also prove to be very important. It is likely that if this predictive algorithm were used by doctors, it would be as a 'second opinion' that could aid their decision in making a diagnosis or recommending a treatment. As such, a doctor may want to be able to interpret the

algorithm's reasoning process in making its prediction. Greater interpretability would also aid in explaining such a prediction and the resulting recommendations to the patient. In terms of interpretability, a single decision tree may be best as it very clearly lays out a series of rules for the predictions it makes. The doctor can then use their judgement to decide if these rules are reasonable. Some accuracy is sacrificed when using a single DT though, as discussed above. Another approach may be to make a prediction using both a single decision tree for interpretability and an SVM for greater accuracy. If the predictions agree, there is no ambiguity. If they don't, the doctor can use their discretion about how to interpret the results or use them to make a diagnosis.

Some further directions for future research might be to investigate the possibility of improving predictive accuracy even further, perhaps experimenting with neural networks deeper and more flexible than the multi-layer perceptron. It is possible that a model with greater capacity may improve predictive accuracy or may lead to overfitting and poor performance on a test set. Further empirical tests may be useful in determining which is the case for this particular data. Another option to explore in terms of improving accuracy is the use of other ensemble models or stacking of models. Perhaps XGBoost may improve on the accuracy of the random forest, or an ensemble of weighted NNs may improve on the multi-layer perceptron.

A further possibility is to collect data from more patients, expanding the scope of the dataset. As the Pima set only contains data from a particularly narrow population – Pima women – its generalisability to other populations is questionable. It may be possible to train a more accurate model using more varied data from men, women and children of varied ages, ethnicities, socioeconomic backgrounds, geographical locations and so on, which could be used to make more accurate predictions for a wider range of new patients.

Reflection

The most important thing I feel I have learned throughout this project is that there is not a huge difference in accuracy between different basic machine learning models when applied without any sort of tuning. Among the top 5 performers: 5NN, NB, MLP, SVM and RF, the difference in accuracy on the CFS set was only about 2%. Perhaps the improvements in accuracy depend on tuning hyperparameters and on the nature of the application. This also makes other factors like interpretability more prominent when choosing an algorithm. This project also clearly demonstrated that it is often possible to improve accuracy through good feature selection. Beyond this, I feel that the experience I gained in implementing certain algorithms from scratch was very valuable, giving me a greater understanding of their inner workings.