COMP3308 A2 report no id

Introduction to Artificial Intelligence (University of Sydney)

COMP3308 Assignment 2 Report

1. Introduction

Better medical technology saves lives and machine learning is a powerful tool that can help aid this technology.

The aim of the study is to understand how we can use machine learning to build models that identify patterns of health risks to different diseases. Machine learning classifiers can infer a person's health status and predict population health risks from several to thousands of attributes such as, BMI, insulin concentration, age etc, which may be difficult for humans alone to correlate with a disease whereas a computer can do this within minutes or seconds. This, in turn, can assist doctors in detecting and treating diseases early and lead to better health outcomes for patients.

Faster medical diagnosis means higher chances of curing or even preventing diseases. Since machine learning in healthcare saves lives, this study is important for better human quality of life and civilization.

2. Data

The data we are using for this study is the **Pima Indians diabetes** database, containing a cohort of all female patients of at least 21 years old. All participants of this study originate from a native American tribe of Pima Indian heritage.

This dataset was originally owned by the National institute of diabetes and digestive and kidney diseases with the donor of database Vincent Sigillito research centre, RMI group leader applied physics laboratory, in 9th May 1990.

The **pima-indian-diabetes** dataset was used for the ADAP learning algorithm to predict the onset of diabetes in 1988. It is also used for a machine learning challenge available on Kaggle, for people who want to practice building a machine learning model. And the data is modified for COMP3308 study purposes, all the missing values were replaced by average value, and all classes changed to normal values.

This database contains **768 instances** with **9 attributes** include:

1. Number of times pregnant
2. Plasma glucose concentration, a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure ($mmHg$)
4. Triceps skin fold thickness (mm)

SIDs:

5. 2-Hour serum insulin ($\mu U/mL$)

6. Body mass index (kg/m^2)

7. Diabetes pedigree function

8. Age (years)

9. Class variable "yes" or "no", where "yes" is interpreted as tested positive for diabetes

There are more instances of “no” in this database where class value “yes” has 268 instances and “no” has 500 instances.

2.1 CFS

Correlation based Feature Selection is an algorithm that finds the best subset of attributes, with a heuristic search strategy that defines how well the attribute can predict the class and the correlation of the attribute with other attributes. The purpose of using the CFS algorithm is to reduce the redundancy in the dataset in order to improve the accuracy of the classifier.

The way **Weka** implements CFS is using the class **CfsSubsetEval** in **weka.attributeSelection** which computes the correlation between attributes and evaluates the worth of a subset of attributes, it returns attributes that are highly correlated with the class and has low intercorrelation. The calculation is done in method **correlate()**, **num_num(), num_nom2() and nom_nom()** depending on attribute types; the methods take in two values and return a float as correlation.

The attributes selected by CFS are:

1. Plasma glucose concentration
2. 2-Hour serum insulin (mu U/ml)
3. Body mass index (weight in kg/(height in m)^2)
4. Diabetes pedigree function
5. Age (years)

SIDs:

3. Results

	No Feature Selection (NFS)		CFS	
	Accuracy (%)	**Time (ms)**	**Accuracy (%)**	**Time (ms)**
ZeroR	65.10%	< 10	65.10%	< 10
1R	70.83%	10	70.83%	< 10
1NN (IBK)	67.84%	< 10	69.01%	< 10
5NN (IBK)	74.48%	< 10	74.48%	< 10
NB	75.13%	10	76.30%	< 10
Decision Tree (J84)	71.74%	20	73.31%	< 10
MLP	75.39%	240	75.78%	130
SVM (SMO)	76.30%	20	76.69%	10
Random Forest	74.87%	160	75.91%	80

Figure 1. WEKA classifier accuracy

	No Feature Selection (NFS)		CFS	
	Accuracy (%)	**Time (ms)**	**Accuracy (%)**	**Time (ms)**
My1NN	67.96%	120	68.23%	98
My5NN	75.25%	118	75.00%	101
MyNB	74.99%	73	76.68%	48

Figure 2. Our classifier accuracy

The results depicted in *Figure 1* and *Figure 2* show the accuracy of the classifiers from Weka's Implementation and our own implementation respectively using 10-fold cross validation.

The '*Time (ms)*' component was generated from Weka's "*Time taken to build model*" output and is accurate to 10 ms. Our model calculated this using Python's '*time.perf_counter()*' method on the same machine and was rounded to the nearest ms.

SIDs:

4. Discussion

4.1 Effect of Feature Selection

Using **Correlation-based Feature Selection (CFS)**, the following five features that had the most correlation with the class identifier:

Plasma glucose concentration
2-Hour serum insulin (mu U/ml)
Body mass index (weight in kg/(height in m)^2)
Diabetes pedigree function
Age (years)

These five features highly correspond with current medical research as diabetes mellitus often results in high blood sugar/glucose (Kharroubi, A. T., & Darwish, H. M. (2015)). Insulin is also an indicator of diabetes but can vary between type 1 and type 2 diabetes. Type 1 patients often have too little insulin in their bloodstream due to poor pancreatic function, resulting in blood glucose to remain in the blood rather than entering cells. Type 2 patients inversely have elevated levels of insulin in the bloodstream as a result of insulin resistance and therefore will also have high blood sugar levels (Wilcox G. (2005)). BMI and age have also been found to be correlated with diabetes (Narayan, K. V et.al (2007) and Laakso, M., & Pyörälä, K. (1985)). Therefore the selection of these 5 features is sensible and correlates with current research.

It was found that when using **CFS**, Weka classifiers increased in accuracy anywhere from 0-2% which appears relatively small gain but in a medical context, this could assist in correctly diagnosing 15 more people in a pool of 768 which is not insignificant.The **ZeroR**, **1R** and **5NN** classifiers are the only ones had the same accuracy before and after CFS. This would indicate that in terms of Weka's classifiers, there are no significant downsides to using **CFS** on this dataset.

On our **1NN** and **Naive Bayes**, implementation, CFS for did slightly increase the accuracy by 0.27% and 1.69% respectively. However, this was not true for our **5NN** implementation which decreased by 0.25% and further investigation is required to determine why this occurred.

Additionally, it was noted that **CFS** dramatically reduced the training times of each model. For most of the Weka classifiers, the training time approximately halved. This makes logical sense as there are 5 features being evaluated instead of 8 which means some operations are doing ~37.5% less work. This was not as apparent in our **KNN** classifiers as this is an iterative approach which requires the model to be re-built on

every test instance and the overhead of rebuilding the model may be greater than having more features. Our **Naive Bayes** implementation on the other hand did reduce by 34% in training time as this algorithm only needs to build the model once before classifying any instances.

However, it is important to note that **CFS** assumes that there are features of the dataset that are more correlated to the class than others which may work well for this dataset but for other datasets, many features may have the same correlation and therefore using **CFS** may be detrimental.

Despite this, **CFS** does show to be a promising strategy to reduce the data required to classify an instance while also increasing the accuracy and reducing the model time.

4.2 Weka Classifier Performance

The first Weka classifier tested on the **pima-indian-diabetes** dataset was the **ZeroR** classifier. This classifier works incredibly simply by using the majority class as the predictor for new data. In this dataset, there were 267 'yes' classifications to 500 'no' classifications of individuals with diabetes. This would mean the **ZeroR** would classify 'no' 65.10% of the time assuming that the testing data contains the same ratio of 'yes's to 'no's. As the ratio of each class is maintained in 10-fold cross validation, this is the expected result obtained.

The next classifier used was the **1R** classifier which generates a general rule for a single predictor with the least amount of error. In this dataset, the predictor with the least amount of error was 'Plasma glucose concentration'. As the same predictor is present in the CFS dataset, there is no difference in accuracy between the **NFS** and CFS dataset.

The next classifier used was Weka's **K-Nearest-Neighbour** implementation. This classifier works by taking the *k* nearest points and classifying the new point with the majority class. The two k values used were k=1 and k=5. When comparing the accuracy of k=1 and k=5, a significant increase can be seen of 6.64% and 5.47% for the **NFS** dataset and **CFS** dataset respectively. This indicates that increasing *k* can have a significant impact on accuracy.

The next classifier tested was ***Naive Bayes***. This classifier works through a probabilistic equation based on Bayes' Theorem which assumes that each attribute is independent as a predictor. This algorithm is often very performant and provides one of the best 3rd best prediction accuracy of all of the Weka classifiers tested for the **NFS** dataset and was the 2nd best predictor in the **CFS** dataset.

SIDs:

Decision Tree (DT) was the next classifier tested and this classifier showed average results when compared to the other classifiers with an accuracy of 71.74% and 73.31% for **NFS** and **CFS** respectively.

Similarly to **DT**, the **Multilayer-Perceptron (MLP)** also displayed middle of the road similar results of 75.39% (2nd best in **NFS**) and 75.78% (4th best in **CFS**) . A key observation for this classifier is that it took significantly more time than any other classifier of 240 ms and 130ms for each dataset therefore may not be suitable for larger datasets.

The best classifier tested was **Support Vector Machine (SVM)** which scored an accuracy of 76.30% for **NFS** and 76.69% **CFS** which was overall the best accuracy of all the classifiers. **SVMs** work by generating a hyperplane that subdivides the features of the dataset into each classification. What makes **SVMs** different to perceptrons is that it tries to maximise the margin between the closest instances (known as supporting vectors) and the hyperplane. This reduces the error as the model will be less susceptible to instances near the hyperplane.

Random Forest (RF) was the last classifier tested on Weka and it produced similar results with the **MLP** and **DT** with an accuracy of 74.87% 75.91% for **NFS** and **CFS** respectively.

4.2 Weka vs Our implementation of KNN and Naive Bayes

In Weka and in our own classifier, k=1 and k=5 were tested. Comparing the accuracy of Weka's implementation to our own, Weka's **1NN** performs slightly worse with an accuracy of 67.84% compared to our 67.96%, a difference of 0.12% in the **NFS** dataset. However, in the CFS dataset, Weka had a better accuracy of 69.01% compared to our 68.23%, a 0.78% deficit.

There could be several reasons why the accuracy may differ between the classifiers. One possibility could be attributed to slight differences in the way that the 10-fold stratifications were constructed when built in Weka versus our own classifier. An entry may be in different folds which results in slightly different distances leading to a skewed accuracy. Secondly, in our implementation, when the *k* neighbours have the same ratio of yes's to no's, 'yes' will always be picked; however, this behaviour is undefined in Weka.

For Naïve Bayes, comparing the accuracy of Weka's implementation to our own, Weka's NB performs slightly better than our NB with Weka's accuracy of 75.13% compared to our 74,99%, a difference of 0.14% in the **NFS** dataset. However, in the

CFS dataset, Weka had a worse accuracy of 76.30% compared to our 76.68%, a 0.38% difference. It is also a key observation that our **NB** algorithm was the 2nd most accurate classifier, only losing to **SVMs** by 0.01%. One possibility that may result in the differences between Weka's **NB** and ours is that similarly to **KNN**, if a point is calculated to have the same probability of being 'yes' or 'no', our algorithm will explicitly pick yes however it is undefined in Weka. Furthermore, as the difference in the accuracies are so small, it could possibly be a result of the difference in 10-fold cross validation construction.

The difference of times between Weka and our own classifiers unfortunately cannot be compared as Weka is programmed in Java while our classifiers are in Python and the difference in time may just be a result of the different ways compilation and runtime occurs in these languages.

5. Conclusion

5.1. Findings Summary

In conclusion, the accuracy and performance can vary greatly with classifiers and this can be compounded with changes to the feature selection of the dataset. The varying rate of accuracy can range from up to 10% with **Naive Bayes** and **SVM** being the most accurate. Therefore, picking a performant classifier is very important especially in a medical context where misdiagnosis can have extensive financial and quality of life effects on patients and overall medical industry.

The idea that without any medical knowledge, an algorithm can predict whether or not an individual has a disease or illness with > 75% accuracy is amazing and incredibly promising for the future of AI in the medical space. With further improvements to these algorithms, we could see AI significantly reduce the turnover time to receive diagnostic results from hospitals and pathology clinics. In turn, it could also provide medical staff with greater confidence of their diagnoses and allow them to more easily prioritise their time with other tasks which may require a greater deal of human intervention.

5.2 Future Work Suggestions

SIDs:

In machine learning, accuracy is one of the key things that we are looking for. There are a few things that can affect the accuracy, we should note them in the future, in order to build a better machine learning model.

First, we can look at the method that we implement ourselves in this study, **KNN** and **NB**.

For K-nearest neighbour classifiers, choosing a different k value will lead to a different result, a small k value will have small neighbourhoods, so the classifier will detect any subtle patterns, but it's not always accurate patterns, the noise will have higher influence on our model which potentially makes the predicting result not accurate. And the runtime of building the model with a small k value can be longer as the complexity of the model is higher. On the other hand, choosing a large k may ignore some noisy points and give a more general pattern, but as k goes larger, the broader pattern we will discover(under-fitting), which we want to avoid. The way we choose the k value should depend on how complex of the pattern the model is going to learn. A simple approach is setting k to the square root of the number of observations, and then test other different values of k to compare the performance to find the best k value.

For Naive Bayes, in our case of study, all the features follow a normal distribution, therefore we use Gaussian Naive Bayes which uses the probability density function to calculate the values.If the features don't follow a normal distribution, we use transformation or any other method to convert it to normal distribution. If the features value are text, we can use Multinomial NB, and if the features are binary, we use Bernoulli NB to build the model. And to improve Naive Bayes models, we should remove features that are highly correlated since they're voted twice in the model, and that can affect the performance of our model.

We used a lot of other different methods on Weka to make our models, and each method has its own advantages and disadvantages, and each model gives us slightly different results. There is a method to improve the accuracy using different models we have, the ensemble method. Ensemble method combines multiple trained models to produce higher accuracy. Note that some of the classifiers won't fit this method such as Naive Bayes because we use this method to reduce variance and NB does not have variance to minimise.

Apart from the methods we are implementing the model, data is also a very important factor that determines the result of the model. A way we can improve with the data is by trying to get more samples. The more data we use to train the model, the model can do more comparisons and get less bias, therefore we are more likely to get more reliable results.

Feature selection and data cleaning are strategies we can perform to improve the machine learning model as well, like the CFS algorithm we used in this study, or other methods of selecting best features, we are looking for features have high correlation with the class, and get less variance in our classifier model, with more relevant features selection to train the model, we can avoid the overfitting result more. For data cleaning, we can modify data to get better results for the model, for example dropping bad data such as outlier, or replacing empty value to mean etc, since the better data we feed to the machine, the better the accuracy we will get for the trained model.

For training data, we should balance the data set, in this dataset more instances of "no" than "yes" where class value "yes" has 268 instances and "no" has 500 instances. It's important to balance out the classes otherwise the model might have biassed towards one class.

Another point to reflect in this study case is all the patients are females only, it may limit the use of the model in the real world, since male and female structures are different, therefore some training values may vary and won't be as accurate for males. A possible future research can be the diabetes prediction model for all sexes.

6.Reflection

In this assignment, we learned how to implement our own classifier using **K-nearest neighbours** and **Naive Bayes,** and how to use Weka to perform different types of classifiers and then evaluate the performance using 10-fold stratified cross-validation. This showed us how accurate different classification models are and how we can use these to make predictions on unseen data.

We also learned ways to improve the model such as **Correlation-based Feature Selection(CFS)**, which allowed us to not only improve the accuracy of our models but also reduce the data used and time taken to build each model. The knowledge of other models also gave us insight into how other models may also approach the same problem while also getting better performance which was incredibly insightful.
It's a great start for us, and we are looking forward to learning more in this area.

7.Reference

Pima Indians Diabetes Database Kaggle:
https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

CfsSubsetEval javadoc:
https://weka.sourceforge.io/doc.dev/weka/attributeSelection/CfsSubsetEval.html

CfsSubsetEval source code:
https://github.com/Waikato/weka-trunk/blob/7b19faa2980f9a7f4c3de0f10f1e75c8f1cb52b1/weka/src/main/java/weka/attributeSelection/CfsSubsetEval.java#L748

Kharroubi, A. T., & Darwish, H. M. (2015). Diabetes mellitus: The epidemic of the century. World journal of diabetes, 6(6), 850–867.
https://doi.org/10.4239/wjd.v6.i6.850

Wilcox G. (2005). Insulin and insulin resistance. The Clinical biochemist. Reviews, 26(2), 19–39.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1204764/

Narayan, K. V., Boyle, J. P., Thompson, T. J., Gregg, E. W., & Williamson, D. F. (2007). Effect of BMI on lifetime risk for diabetes in the US. Diabetes care, 30(6), 1562-1566.
https://diabetesjournals.org/care/article/30/6/1562/30745/Effect-of-BMI-on-Lifetime-Risk-for-Diabetes-in-the

Laakso, M., & Pyörälä, K. (1985). Age of Onset and Type of Diabetes.
https://doi.org/10.2337/diacare.8.2.114

"Learn Naive Bayes Algorithm | Naive Bayes Classifier Examples", 2022
https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

SIDs: