2018

COMP3308 REPORT

450412950, 470405196

SECTION 1 - AIM

This study was conducted for a few purposes. Firstly, the aim was to understand how to code and implement KNN / Naive Bayes on a large data set (Pima Indian Diabetes) and compare this to WEKA's in-built functions. Through this study, the WEKA tool was used to help understand how to manipulate datasets and retrieve statistical information through software-based methods. Secondly, another goal was to produce the results in a statistically meaningful way and be able to draw clear conclusions from the data.

More holistically, the study is important in understanding the accuracy of different classification algorithms and the variations between methods on different datasets, including the advantages and disadvantages of using each classification algorithm and the relevance of Correlation-based Feature Selection. Our initial hypothesis was that having more initial data attributes would produce a more accurate result.

SECTION 2 – DATA

In this assignment, our dataset was based off 'Pima Indian Diabetes', where a study was conducted in 1988 on diabetes and digestive and kidney diseases. The study was conducted on vast populi with a Pima Indian heritage however, for the sake of this assignment the dataset was modified such that all instances of our given dataset are females of at least 21 years old and are of the Pima Indian heritage. All instances possess attributes:

- Number of times pregnant
- Plasma glucose concentration in 2 hours in an oral glucose tolerance test
- Diastolic blood pressure (mm Hg)
- Triceps skin fold thickness (mm)
- 2-Hour serum insulin (mu U/ml)
- Body mass index (weight in kg/(height in m)^2)
- Diabetes pedigree function
- Age (years)
- Class variable is_diabetic("yes" or "no")

For the sake of our implementation of 'Kth Nearest Neighbour', 'Naive Bayes', computation of CFS datasets and 10-fold stratified cross-validation algorithms, we created a class

'Person' which had congruent attributes listed above, and in addition, a variable to hold the Euclidean distance between two instances, and an array of floats which was utilised to find the Euclidean distance and computation of CFS datasets.

A good dataset is one that contains attributes with strong correlation to the class, but low correlation with other attributes. As such, Weka's CFS function attempts to choose a subset of the original attributes that have the best predictive ability to the classifier and the lowest correlation to other attributes.[1] In the Pima diabetes database, WEKA identified the following attributes to satisfy these criteria, and omitted the other attributes. Intuitively, this makes sense as these chosen attributes have a relatively low correlation with each other, but a higher degree of correlation to the is_diabetic classifier (e.g. it makes sense that BMI has some effect on whether the patient is diabetic).

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

SECTION 3 – RESULTS AND DISCUSSION

Weka is a machine-learning program with a set of in built features and algorithms, and the program can be used effectively for data science, data mining and machine learning tasks. In this study, a range of Weka functions were used including the Preprocessing feature to normalise the data, the Classify function to run the algorithms and Select Attributes fun to reduce the amount of attributes using Correlation-based Feature Selection (CFS). Ultimately, WEKA played an important role in collating results and measuring the accuracy of the different algorithms.

Section 3.1 - Overall Performance of CFS vs original dataset

Weka Algorithms

	ZeroR	**1R**	**1NN**	**3NN**	**NB**	**DT**	**MLP**	**SVM**

[1] https://www.cs.waikato.ac.nz/~mhall/thesis.pdf

No CFS	65.1042 %	70.8333 %	67.8385 %	72.6563 %	75.1302 %	71.875 %	75.3906 %	76.3021 %
CFS	65.1042 %	70.8333 %	69.0104 %	73.3073 %	76.3021 %	73.3073 %	75.7813 %	76.6927 %

The above table is a summary of running the required algorithms in Weka, on both the original Pima and Pima-CFS datasets, using stratified 10 fold cross validation. All algorithms except the first two had a higher degree of accuracy when using the CFS dataset rather than the original dataset. The two exceptions were:

- ZeroR**:** Running ZeroR on the CFS and original dataset produced the same accuracy of 65.1%. This result is explained by the nature of the ZeroR algorithm, which ignores all other attributes except the classifier itself. It simply chooses the classification with the highest likelihood (in this case, is_diabetic = “No” which had 500 instances compared to 268 for “Yes”). As such, running the ZeroR algorithm on both the CFS and original dataset predicted the correct value 500/768 = 65.1% of the time. This is taken as the benchmark level of accuracy; all other classification algorithms are expected to have an accuracy rate better than this.[2]
- 1R: The 1R algorithm demonstrated the same accuracy for the CFS and original dataset as it models the predictor off a single attribute, shared between both datasets. In this case, it chose Column B as it produced the smallest total error compared to all other predictors and arrived at a 70.8% accuracy (predicting 544/768 instances correctly in both cases).

For the other algorithms, the CFS dataset produced marginally better results (no more than 1-2% better in absolute percentage terms). Overall, CFS was beneficial and yielded slightly more accurate results.

Section 3.2 - Comparisons between Classifiers

In terms of accuracy, the classifiers performed in this order on the original dataset (in descending terms of accuracy):

1. SVM

[2] http://chem-eng.utoronto.ca/~datamining/dmc/zeror.htm

2. MLP
3. Naive Bayes
4. 3NN
5. Decision Tree
6. 1R
7. 1NN
8. ZeroR

Whilst it was noted that ZeroR should represent a baseline minimum level of accuracy for classifiers, and 1R a slightly more accurate version of this, it was interesting to see that running the 1 Nearest Neighbour algorithm only produced 67.8% accuracy (and 69% on the CFS dataset). Whilst the K Nearest Neighbour algorithm is simple to understand and implement, its relatively less accurate result may be attributed to the fact that KNN equally weights all data points in the Euclidean distance calculation, and therefore may overweight the importance of attributes which have less correlation to whether the patient is diabetic; the choice of '1' nearest neighbour made this risk even greater. Conversely, 3NN (72.6% accuracy) performed better than 1NN as the presence of more points of reference eliminated the sole exposure to a single 'nearest' data point. As such, there was less risk of outliers or single data points that did not give an accurate classification. An additional disadvantage of using the KNN method is that the distance of the input points to all the points in the original dataset needs to be calculated, and the running time can therefore be quite slow.[3] Furthermore, the optimal value of 'K' needs to be chosen to produce the most accurate result, and this often may be difficult to determine.

Naive Bayes was more accurate than the K Nearest Neighbour algorithm, with accuracy ranging from 75-76% (for original vs CFS datasets). This may be attributed to the fact that whilst KNN equally weights all data points in its Euclidean distance calculation, Naive Bayes does not and weights attributes based on their probability of occurrence. Furthermore, Naive Bayes is a relatively fast algorithm that can work well with large datasets, and can scale linearly as the amount of attributes is increased. [4] However, a drawback of Naive Bayes assumes independent attributes and this can be less effective when the attributes have some degree of correlation with each other.

DT: The DT algorithm followed the general trend that the CFS dataset produced more accurate results than the original dataset (73.3% vs 71.8%). This can be explained by the

[3] http://www.cs.upc.edu/~bejar/apren/docum/trans/03d-algind-knn-eng.pdf
[4] https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_nb.htm#BABIIDDE

process of decision trees and their tendency to over-fit when using datasets with complex and multiple attributes, thus the CFS produced a more accurate result as it only selected certain Pima attributes that were most correlated to is_diabetic.[5] The algorithm works by splitting the datasets into a squares of desired values that follow a recognisable trend, hence reaching a definitive classifier (i.e. is_diabetic) when reaching a leaf of the DT.[6]

Accuracy can be improved by pruning (i.e. taking away unmeaningful attributes of a dataset to reduce the complexity of each iteration in a tree). This explains why the CFS dataset (which only took 5 of the original 8 attributes) returned more accurate results than the original unmodified dataset and why the overall results returned mediocre accuracy in comparison to the other algorithms - our classifiers were dependent on a considerable amount of complex numerical variables.

MLP: The discrepancy between accuracy results for the original Pima and CFS datasets using the MLP algorithm was negligible - being approximately <0.4%. Additionally, MLP was one of the most accurate algorithms we tested on Weka and this can be explained by analysing the complex process of MLP algorithms.

The MLP algorithm works by running a dataset through multiple layers of nonlinear functions to achieve an overall nonlinear trend between the classifiers. The number of function layers typically depends on the complexity of the classifier plot division, which is usually determined by how vast the numerical attributes of a dataset is. By running the dataset through more functions, the MLP algorithm is able to predict a more fitting plot division by adhering to finite curves within the division. The advantage of using an MLP algorithm is that you can use either data set and still attain similar accuracies, thus explaining why our unmodified and CFS dataset accuracy differ less than any other algorithm. The disadvantages of using this algorithm lies within the runtime as it exponentially worsens as more functions are added. However in conclusion, especially when dealing with large datasets, the advantages outweigh the disadvantages if the runtime is not of concern.[7]

SVM:

The computation of SVM is very similar to MLG which can explain why the two datasets yield similar trends in its accuracies between CFS and our unmodified dataset, also being

[5] https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052
[6] https://www.mindtools.com/dectree.html
[7] http://deeplearning.net/tutorial/mlp.html

<0.4%. However, the difference in computation is instead of plotting the points on a 2-dimensional plane and finding a nonlinear classifier division, it uses a 3-dimensional hyperplane and vector dot products to separate the classifier plot division and as such, will provide a more overall accurate representation. This explains why accuracy discrepancy between our unmodified dataset and CFS dataset with both the algorithms are similar, but SVM algorithm obtains a superior overall accuracy. Just like MLG, in most cases SVM will be the most ideal accuracy test, especially over 10 folds. The only disadvantage is computation time as the algorithm utilises a number of complex mathematical functions.[8]

Section 3.3 - Our Implementations of KNN and NB

	My 1NN	My 3NN	MyNB
No CFS	68.6731%	0.7428	0.7507
CFS	67.2843	0.7493	0.7582

Running our own 3NN and Naive Bayes algorithms dataset yielded slightly better results on the CFS dataset, whereas the 1NN algorithm produced slightly better results on the original dataset. This shows that whilst CFS may be advantageous in reducing less correlated or relevant features, it is not in all circumstances guaranteed to produce a better result due to various anomalies in different datasets. Another observation was that using a stratified 10-fold method was advantageous compared to using a simple test/training ratio over 1 iteration. This is because all 768 instances are eventually tested over the 10 iterations, and the average accuracy gives a much better indication than randomly selecting a test sample.

Overall, our own implementations of the algorithms produced very similar results to the same algorithms run on Weka (although 3NN was more accurate than the Weka one by around 1%), demonstrating a successful implementation of the two algorithms. The slight discrepancies in accuracy may be attributed to slight differences in the implementations of the algorithms, taking into account factors such as rounding conventions.

SECTION 4 – CONCLUSIONS AND FUTURE WORK

[8] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Through this study, we experimented with our implementation of the Naive Bayes and Kth Nearest Neighbour algorithms, and found that they produce results to a mid-high level of accuracy of around 68-75%. Whilst this result is fairly solid for relatively simple algorithms, there is definitely room to improve the classification accuracy by exploring alternative algorithms such as SVM, Decision Trees and MLP. We were able to observe the differences in accuracy between the different classification methods, i.e. Naive Bayes was more effective than KNN, and the most accurate algorithms overall were SVM, MLP and Naive Bayes. It was important that the dataset was initially normalised before it was run through our code and Weka's implementation, so that attributes with higher mean values were not unnecessarily over weighted.

A secondary conclusion was that using Correlation-based Feature Selection on datasets has a small, but noticeable impact on improving the accuracy of the algorithms. This disproves the initial hypothesis, which was that more data attributes would produce a more accurate result. The relative accuracy of CFS was highly consistent across the vast majority of all test cases. Furthermore, CFS was beneficial in reducing the size of the dataset by reducing the amount of attributes, and has a positive impact on runtime of the code's implementation.

Future Suggestions

A direct extension of this exercise would be to apply a similar study to other machine learning algorithms and evaluate the accuracy accordingly, and ultimately find the most relevant and scalable algorithm for both numerical and categorical datasets.

On a larger scale, a future suggestion could be to extend this study to a much grander scale - i.e. the diagnosis of diseases (physical or mental) spanning across a vast dataset of symptoms, attributes and diagnoses. In essence, this would involve extremely large historical datasets of patients with differing characteristics and classifications. Ultimately, the goal would be to use data science and machine learning methods to diagnose future patients more accurately and both to reduce the amount of misdiagnoses and make medical services more accessible to the extended population. This is an example of how the study of machine learning and data science can positively impact society and bring the medical field more in line with current technological advancements.

SECTION 5 – REFLECTION

More data points are not necessarily always better

Whilst we initially assumed that the presence of more seemingly relevant attributes would overall give a better result, this was not necessarily true. This was evident by observing the better performance of the CFS vs original dataset. Furthermore, with classifiers such as Naive Bayes, it is the presence of relevant, *uncorrelated* attributes that produces the best result. Likewise with KNN, whilst 3NN was more accurate than 1NN, there comes a turning point where increasing the value of K no longer produces a more accurate result. This shows that whilst machine learning is premised on having access to a significant quantity of data, there is still the inherent trade-off between data quality and quantity.

Small rounding errors can lead to significant discrepancies

In our experience of implementing the code, at first we neglected to realise the importance of rounding numbers to an almost microscopic level of detail, and spent many hours trying to work out why the code would not pass the test cases in terms of accuracy (e.g. obtaining 93% compared to the required 95%). In some cases, this was a simple matter of rounding numbers with floats instead of integers; we found that minute errors like this, applied over large datasets, can cause significant classification discrepancies. Errors like this throughout the code made us realise how important it was to plan the construction of your methods and data allocation before implementing your code, as it can be problematic threading out logic errors between a considerable amount of complex code.

This exercise correlates to real world software development

Through doing this exercise, we were exposed both to writing the algorithmic implementation in Python and documenting the results and summarizing findings. This process correlates to real world software development in the sense that all software developed eventually needs to be explained to users and various stakeholders. As such, documenting the results and presenting the findings in a clear and concise way gave us experience to build upon in the future.