COMP3308 -Assignment 2 Report

SID: 530297879, 530325264

1. Introduction

The aim of this study is to show how we can use machine learning to build models to identify patterns and predict outcomes from the sensors and medical data of two different classification datasets: **Pima Indian Diabetes** and **Room Occupancy**. We investigate how accurate it is to predict the likelihood that a person has diabetes using health-related factors such as BMI, insulin level, age, etc., and whether it is possible to estimate room occupancy using sensor readings such as temperature, light, etc.

This study is important because it demonstrates the practical applications of data and machine learning, by highlighting how machine learning can solve real-world problems in both health and environmental monitoring. This can assist doctors detect and treat the disease earlier and lead to better outcomes for patients, potentially saving their lives. Similarly, reliable room occupancy detection can also lead to a smarter building and energy savings environment.

2. Data

2.1 Pima Indian Diabetes Dataset

This dataset consists of all female patients from a native American tribe of Pima Indian heritage under the age of at least 21 years.

The dataset was originally owned by the National Institute of Diabetes and Digestive and Kidney Disease with the donor of the database: Vincent Sigillito Research Center, RMI Group Leader, Applied Physics Laboratory, established in May 1990.

The **pima-indian-diabetes** dataset was used for the ADAP learning algorithm to forecast the onset of diabetes mellitus in 1988 (Kaggle). It inspires people to build a machine learning model to predict whether patients have diabetes or not as a challenge. This dataset was modified for 'COMP3308 study analysis', by replacing missing values with the average class, changed to nominal values, and a row of column names was added. It is licensed under CC0: Public Domain. Normalise with Weka to 'pima.csv'

The dataset contains **768 instances** with **9 attributes** (8 numeric and 1 nominal).

Attributes include:
- **tp** (`integer`): Number of times pregnant
- **gc** (`integer`): Plasma glucose concentration, 2 hours in an oral glucose tolerance test
- **bp** (`integer`): Diastolic blood pressure ($mm\ Hg$)
- **sft** (`integer`): Triceps skin fold thickness (mm)
- **si** (`integer`): 2-Hour serum insulin ($\mu U/ml$)
- **bmi** (`float`): Body mass index (kg/m^2)
- **dpf** (`float`): Diabetes Pedigree Function
- **age** (`integer`): Age ($years$)
- **class** (`string`): Class variable ("*yes*" or "*no*"), where "*yes*" is positive for diabetes and "*no*" is negative for diabetes

Class distribution: - "*no*": 500 instances - "*yes*" 268 instances (∼65% negative and ∼35% positive)

2.2 Room Occupancy Dataset

This dataset consists of instances; each corresponds to a point in time between December 2017 and January 2018 for a period of 4 days. It records the environmental factors in a controlled manner with the occupancy in the room having between 0-3 people at the time.

The dataset was orignally owned by Adarsh Pal Singh, Vivek Jain, Sachin Chaudhari, Frank Alexander Kraemer, Stefan Werner, and Vishal Garg, "Machine Learning-Based Occupancy Estimation Using Multivariate Sensor Nodes", in 2018 IEEE Globecom Workshops (GC Wkshps), 2018. The dataset is then donated to the UCL machine learning repository on 15 August 2023.

The **room occupancy** has no missing values and was modified for 'COMP3308 study analysis'. It is licensed under Creative Commons Attribution 4.0 International. Normalise with Weka to 'occupancy.csv'.

The dataset contains **2025 instances** with **5 attributes** (4 numeric and 1 nominal).

Attributes include:
- **temp** (`float`): Temperature reading (°C)
- **light** (`integer`): Light reading (Lux)
- **sound** (`float`): The output of the amplifier attached to a microphone ($Volts$).The louder the sound, the higher the voltage.
- **CO2** (ttinteger): Carbon dioxide reading (PPM)
- **class** (`string`): Class variable ("*yes*" or "*no*"), where "*yes*" is 1-3 people in the room and "*no*" is no people in the room.

Class distribution: - "*no*": 1646 instances - "*yes*" 379 instances (∼18.7% negative and ∼81.3% positive)

Similarities:

Both Pima Indians Diabetes and Room Occupancy datasets are used for binary classification tasks with class labels of "yes" and "no". Each dataset contains only numeric attributes (excluding the class label) and no missing values are present in both datasets (after preprocessing). Both datasets require data cleaning due to differing feature scaling, and each of the instances represents a real-world measurement scenario that is used to make the prediction.

Differences:

Pima Indians Diabetes dataset is health-related and has a total of 9 attributes, while the Room Occupancy dataset belongs to a smart environment domain and has only a total of 5 attributes. The Room Occupancy dataset has more number of instances, containing 2025 instances compared to Pima Indian Diabetes, which contains only 768 instances. Both datasets have their own specific practical application, and got different class distributions (number of instances "yes" to "no" ratio).

3. Results and Discussion

3.1 Results

	ZeroR	1R	1NN	7NN	NB	DT	MLP	SVM	RF
Diabetes	65.10%	70.83%	67.84%	74.35%	75.13%	71.74%	75.39%	76.30%	75.87%
	<0.01 s	0.01 s	<0.01 s	<0.01 s	0.01 s	0.06 s	0.50 s	0.03 s	0.17 s
Occupancy	81.28%	98.47%	99.41%	99.21%	96.74%	99.46%	99.26%	98.42%	99.65%
	<0.01 s	<0.01 s	<0.01 s	<0.01 s	<0.01 s	0.01 s	0.77 s	0.02 s	0.2 s

Table 1: Classifier accuracy and time comparison using Weka classifiers

	My1NN	My7NN	MyNB	MyEns
Diabetes	69.02%	74.47%	74.99%	75.39%
	2.452 s	2.465 s	1.817 s	3.693 s
Occupancy	100.0%	99.26%	96.84%	99.26%
	8.819 s	9.129 s	4.857 s	13.91 s

Table 2: Classifier accuracy and time comparison using own implementations

3.2 Discussion

3.2.1 Classifier Performance of all classifiers

1. **ZeroR:** On **pima.csv** dataset, the model correctly classifies 500 instances and incorrectly classifies 268 instances. On **occupancy.csv** dataset, the model correctly classifies 1646 instances and incorrectly classifies 379 instances. This resulted in the ZeroR classifier's accuracy being 65.10% and 81.28% respectively, of the time in the full training set, the lowest accuracy out of all classifiers performed in both datasets. However, the time taken to build both models is really fast, approximately < 0.01 seconds.

2. **1R:** On **pima.csv** dataset, the model correctly classifies 544 instances and incorrectly classifies 224 instances. This resulted in the accuracy of the OneR classifier being 70.83% of the time in the full training set, close to the average accuracy of all classifiers performed in this dataset. On **occupancy.csv** dataset, the model correctly classifies 1994 instances and incorrectly classifies 31 instances. This resulted in the accuracy of the OneR classifier being 98.47% of the time in the full training set, a high performance accuracy. The time taken to build both models is really fast, approximately $\leq$ 0.01 seconds.

3. **K-Nearest Neighbours:** This classifier works by taking the k nearest points and classifies the new point with the majority class. The k values used were $k = 1$ and $k = 7$. For both Weka and our own Python implementation, an increase in accuracy can be seen for 1NN and 7NN: 67.84% , 74.35% respectively for Weka and 69.02% , 74.47% respectively for our own Python implementation on the **pima.csv** dataset. The model correctly classifies 500 instances and incorrectly classifies 268 instances. The time taken to build both 1NN and 7NN models in Weka are < 0.01 seconds. However, the time taken to build 1NN and 7NN in our own Python implementation is approximately ~2.5 seconds.

On **occupancy.csv** dataset, the difference in accuracy for both Weka's and our own Python implementations 1NN and 7NN are not significant: 99.41% , 99.21% for Weka and 100.00% , 99.26% for our own Python implementation. The time taken to build both the 1NN and 7NN models in Weka are < 0.01 seconds. However, the time taken to build 1NN and 7NN in our own Python implementation is approximately ∼9 seconds.

Overall, our own implementation performance has higher accuracy but longer building time compared to Weka performance.

4. **Naive Bayes:** This classifier works through a probabilistic equation based on Bayes Theorem. For both Weka and our own Python implementation, there was not that much accuracy difference in **pima.csv** dataset and **occupancy.csv** dataset. Diabetes: 75.13% for Weka and 74.99% for our own Python implementation. Occupancy: 96.74% for Weka and 96.84% for our own Python implementation. However, the time taken to build this model in Weka is very fast compared to our own implementation: $\leq$ 0.01 to 1.817 seconds in diabetes and 4.857 seconds in occupancy.

5. **Decision Tree:** On **pima.csv** dataset, the model correctly classifies 551 instances and incorrectly classifies 217 instances. On **occupancy.csv** dataset, the model correctly classifies 2014 instances and incorrectly classifies 11 instances. This resulted in the accuracy of the Decision Tree (J48) to be 71.74% and 99.46% respectively, of the time in the full training set. The time taken to build this model is approximately 0.06 and 0.01 seconds, respectively.

6. **Multilayer-Perceptron:** On **pima.csv** dataset, the model correctly classifies 579 instances and incorrectly classifies 189 instances. On **occupancy.csv** dataset, the model correctly classifies 2010 instances and incorrectly classifies 15 instances. This resulted in the accuracy of MLP being 75.39% for Diabetes, and 99.26% for Occupancy, of the time in the full training set. The time taken to build this model is the longest compared to all models in Weka, approximately 0.50 seconds for Diabetes and 0.77 seconds for Occupancy.

7. **Support Vector Machine:** This classifier (SMO in Weka) almost has prediction similar to the accuracy of MLP and is the best classifier tested for **pima.csv** dataset: 76.30% on accuracy and 0.03 to build this model. For **occupancy.csv** dataset: 98.42% on accuracy and 0.02 seconds to build this model.

8. **Random Forest:** This classifier also produces similar results. **pima.csv** dataset: 75.87% on accuracy and 0.17 to build this model. It has the best classifier tested for **occupancy.csv** dataset: 99.65% on accuracy and 0.2 seconds to build this model.

9. **MyEssemble:** This classifier is from our own Python implementation, a combination of 3 classifiers: two Nearest Neighbors (with different k) and one Naive Bayes. It combine predictions by taking the majority vote. The model produces an accuracy of 75.39% , 99.26% for **pima.csv** and **occupancy.csv** dataset respectively.

3.2.2 Comparison with Weka

For k-Nearest Neighbours, our model achieved slightly higher accuracy compared to Weka in both **pima.csv** and **occupancy.csv**. Naive Bayes produced an identical result with a difference ∼0.2% between the Weka and Python implementation model. All of

Weka's running times are way less than our models and are not greatly affected by the number of values in the dataset which leads us to believe that they use a more efficient algorithm. However, our classifiers result mainly in a similar / higher accuracy than Weka.

3.2.3 Comparison of Ensemble and Individual Classifiers

MyEns consistently performs the best or close to the best when compared to the other classifiers individually in terms of accuracy;
Diabetes -> **MyEns**: 75.39%, **My1NN**: 69.02%, **My7NN**: 74.47%, **MyNB**: 74.99%.
Occupancy -> **MyEns**: 99.26%, **My1NN**: 100.00%, **My7NN**: 99.26%, **MyNB**: 96.84%.

This shows that the ensemble voting helps mitigate the weaknesses of each classifier and leverages their strengths. The only downside to MyEns is that it has the longest running time of all classifiers. This is because it combines the predictions of all three classifiers, which means that it has to run all three classifiers.

3.2.4 Comparison of Dataset Performance

In all cases, each classifier performed better with the **Room Occupancy** dataset compared to the **Pima Indian Diabetes** dataset in terms of accuracy. This is most likely because the diabetes dataset has more attributes, which increases complexity and is more likely to have noisy attributes that can affect the performance of the classifiers. It also makes sense why all classifiers did better with the **Room Occupancy** dataset which has more rows. Having a larger dataset will lead to a larger training set as we are using stratified cross-validation folds; this means the classifiers have more examples to learn from and can perform better. The results also show that the classifiers have more running time when classifying the **Room Occupancy** dataset which is reasonable as it would require more time to classify a larger dataset.

4. Conclusion

4.1 Result Summary

Overall, the ensemble classifier MyEns consistently outperforms the individual classifiers (My1NN, My7NN, MyNB) across both datasets showing that ensemble voting is more effective than individual classifiers even though it would have a longer running time than the others. Comparisons with Weka's classifiers also show that our classifiers lack efficiency, as all Weka's classifiers have a faster running time than ours. It is also reasonable that each classifier performed better with the **Room Occupancy** dataset compared to the **Pima Indian Diabetes** dataset due to it having less attributes and more rows.

For **Pima Indian Diabetes** which has a smaller dataset, the best Weka classifier is SVM (Support Vector Machine) which has an accuracy of 76.30% and running time of 0.03 seconds, while the best classifier of our own implementation is the Ensemble model with an accuracy of 75.39% and a running time of 3.693 seconds. For the **Occupancy** dataset which has more rows, the best Weka classifier is Random Forest (RF) which has an accuracy of 99.65% and running time of 0.2 seconds, while the best classifier of our implementation is 1NN with an accuracy of 100% and a running time of 8.819 seconds.

4.2 Future Work Suggestions

Based on the results, our classifiers have proved useful in classifying simple datasets that have been pre-processed. This means that we can rely on these classifiers to perform

other analysis and predictions on unseen data, provided that we have a training set. The results have also shown that our models work better with smaller datasets, meaning that we can implement feature selection to future datasets to get the most important attributes and reduce noisy data. This will also reduce overfitting and speed up training time.

For k-Nearest Neighbour (KNN), we can improve it by implementing hyperparameter tuning, which explores automated tuning of the k-value which will improve the performance of the model. Lower k-values are affected by noisy data and may not detect accurate patterns, while larger k-values may have a broader patterns, which can result in underfitting. The optimal k-value depends on how complex the pattern of the model is.

For Naive Bayes (NB), we can use the simple approach of using a probability density function because all the features follow a normal distribution. However, this is not always the case for other datasets. In order to improve this model, we can use other forms of NB for other types of data. For nominal features, we can use Multinominal NB and for binary features, we can use Bernoulli NB. We can also improve the model further by removing attributes that have a high correlation with each other. Highly correlated features violate Naive Bayes' independence assumption, causing overcounting and reducing prediction accuracy and probability reliability.

5. Reflection

530297879:

In this assignment, I learned how to implement my own classifier using K-Nearest Neighbour and Naive Bayes, and how to use Weka to perform 10-fold cross-validation on the following algorithm: ZeroR, 1R, K-Nearest Neighbours, Naive Bayes, Decision Tree, Multi-Layer Perceptron, Support Vector Machine, and Random Forest. This shows how accurate different classification models are and how this might be used in the future to make predictions on unseen data. Before this project, I didn't understand why combining classifiers could perform better than individual ones. But after implementing and testing it with our own Python MyEns code, I saw the benefit (better accuracy) and weakness (longer build time). Overall, this project deepened my understanding of machine learning and highlights the importance of good model selection and the use of it for real-world applications.

530325264:

Throughout this assignment, I have learned how to make a classifier program that can predict data based on other examples from scratch. This provides a lot of insight for me on machine learning techniques, especially k-Nearest Neighbours, Naïve Bayes and Ensemble methods. I understood more of how each algorithm functions and how they make predictions based on distance or probability. This assignment also taught me the importance of pre-processing as well as stratified cross-validation folding the dataset before moving on. I also learned how different classifiers perform differently based on a dataset's number of attributes and size from measuring the performance using evaluation metrics like time and accuracy. Overall, the project enhanced both my theoretical knowledge and practical skills in model building, evaluation, and analysis, giving me a deeper understanding of applied machine learning.