MLDM 2021 Coursework Group: *The Mean Squares*

Danylo Kovalenko (6413526) DK00266@SURREY.AC.UK
Emily Gould (6660230) EG00823@SURREY.AC.UK
Jamie Dance (6661320) JD01321@SURREY.AC.UK
Jenna Wilkes (6662507) JW02077@SURREY.AC.UK
William Hemsley (6414699) WH00189@SURREY.AC.UK

Abstract

This paper compares the application of 5 machine learning algorithms in different experimental setups. Decision Tree, Perceptron, Multi-Layered Perceptron (MLP), Naïve Bayes and Aleph ILP algorithms are explored to find suitable predictors for binary, multi-class and relational problems. The results indicate that the MLP is the best classifier for large datasets whereas the single-layer perceptrons are not suitable for the given problems. Relational learning results suggest that the addition of background knowledge is beneficial and that relational learning algorithms perform best on complex relational datasets.

1. Project Definition

This project explores three datasets with different characteristics to compare various machine learning algorithms. Datasets for rain data, wine quality data and carcinogenesis relational data are analysed and explored in-depth, with the aim of solving their associated learning problem.

The rain dataset [1] is large (145,461 instances) and contains a mixture of categorical and numerical variables. It is a binary classification problem, intending to predict whether it will rain the next day. This data has been accumulated for most days between the years 2008 to 2017, with the information given for 49 locations in Australia. Comparatively, the wine dataset [2] is smaller in size (6,497 instances) and contains 12 numerical features. It contains attributes of Portuguese "Vinho Verde" wine, providing a multi-class classification problem, as wines, both red and white, are classified into one of ten quality classes.

The carcinogenesis dataset [3] explores relational learning with a binary classification problem. The aim is to classify chemicals into those that cause or do not cause cancer (active or inactive). The data provided was acquired through the testing of chemicals on rodents. The dataset contains data for 21 Ashby indicators and their numerical counts, which are properties of molecules that indicate carcinogenicity. There are only 300 instances in this dataset, however, the data represents complex relationships and a rich background knowledge file is provided.

The overall aim of the project is to successfully find the algorithms best suited to each of the problems associated with each dataset. The algorithms to be explored are decision trees, Naïve Bayes, neural networks and ILP. The hyper-parameters of the algorithms will be explored and tuned to aid the classification. The expectation is that each problem will be solved by a different machine learning algorithm due to the distinct differences in datasets such as size and dimensionality. A highly effective classifier should be learned for each dataset, with supporting evidence such as evaluation metrics demonstrating the model competency. The models learnt should avoid overfitting or underfitting the datasets and should generalise effectively with new data.

Hypotheses were made at a dataset-specific level due to the highly different characteristics of the datasets. Neural network algorithms were expected to perform most effectively with the rain dataset as the dataset has the largest amount of data. It was also expected that the decision tree algorithm would be a poor classifier for the rain dataset as it has high dimensionality. The Gaussian Naïve Bayes was expected to perform well with the wine dataset because the wine dataset contains mainly continuous numerical data. It is expected that a relational learning algorithm, such as the Aleph Inductive Learning (ILP) algorithm will perform best on the carcinogenesis dataset and performance will be improved with the addition of background information. After implementing Aleph, the non-relational algorithms will also be compared and evaluated on the non-relational data in this dataset.

To evaluate the performance of the learning algorithms on the datasets, metrics were gathered once parameters were tuned and finalised. The accuracy of a model represents the fraction of correct predictions and is a good metric when comparing balanced target classes. The 'F1 weighted' score (referred to as 'F1' throughout) accounts for dataset imbalance by calculating the correct predictions *per class* and averaging them. The average is weighted depending on the number of instances in each class. **R**eceiver **O**perator **Curve** (ROC) shows the relationship between the true positive rate and the false positive rate of a classifier. The **A**rea **U**nder the **C**urve (AUC) of the ROC provides an understanding of the performance - the larger the area the better. Using these metrics, it can be assessed, through comparable evidence, the most suitable algorithm for each dataset.

2. Data Preparation

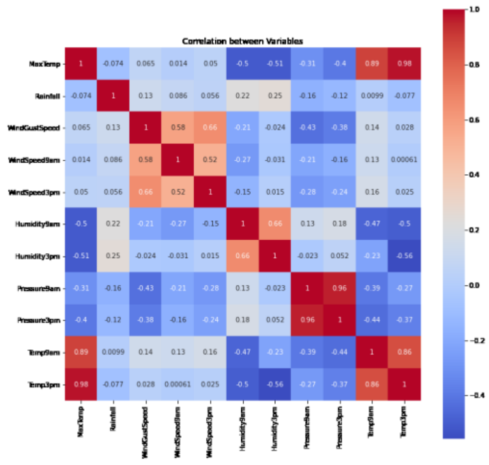
Initially, the wine data consisted of two separate white wine and red wine datasets, sharing the same features. By creating a column for colour, the two datasets were merged creating more examples for each quality class. Some classes were vastly underrepresented, and even after merging there was insufficient data to perform classification for the extreme classes (e.g. 1, 2, 3, 9, 10), so these classes were removed.

After binary encoding the colour feature, the resulting wine dataset contained 13 variables - 2 discrete and 11 continuous. This is far smaller than the rain dataset, which has 23 variables - 7 categorical and 16 numerical - and 145,460 entries. The wine dataset did not contain any null values whereas the rain dataset contained 343,248 (10.3% of the dataset). Notably, 2.25% of entries did not have a classification for the target variable *'Rain Tomorrow*' and such instances were immediately removed. Variables containing over 15% null values were disregarded, leading to the removal of '*evaporation*', '*sunshine*', '*cloud9am*' and '*cloud3pm*'. The remaining null values in each categorical variable were imputed by the mode and null numerical variables were imputed by the mean. The mode/mean were calculated per location (rather than the entire dataset) because Australia is a large country and weather varies by location. This gave more consistent and precise estimates. Certain locations did not have any values for particular variables and in these instances, the mean or mode was applied using every entry for that variable.

Multicollinearity was explored as correlation between variables can affect their statistical significance. Relationships with a correlation value close to 1 or -1 are considered strong, with those close to 0 considered weak. Certain features in the rain dataset showed high multicollinearity (see Fig 1), where darker colour represents a stronger correlation. The values for 9am and 3pm of the *Pressure* and *Temp* variables had a very high correlation with values of 0.96 and 0.86 respectively. To resolve this, these different time features were averaged. *MaxTemp* and *Temp* had near-perfect correlation (0.97), so *MaxTemp* was removed. When exploring the multicollinearity in the wine data, a negative correlation (-0.69) between *alcohol* and *density* was found, so *density* was removed. Similarly, *total sulfur dioxide* and *colour* were also removed.

Categorical data was transformed into numerical values to be interpreted by the learning algorithms. Both datasets displayed evidence of outliers, so it was appropriate to remove and investigate potentially anomalous data. The data was normalised (excluding the target variable) so that normality could be assumed. As 99.7% of the data should lie within three standard deviations from the mean, data that strayed further than 3 deviations from the mean was excluded in an attempt to remove extreme and likely anomalous values. Consequently, 2.83% and 1.62% entries from the wine and rain dataset were removed, respectively.

Figure 1. Heatmap of the correlation between numeric variables in the rain dataset.



For the carcinogenesis dataset, no preprocessing was required for the relational algorithm, however different methods of preprocessing the extracted numerical data was explored (discussed in §4.3.5). Due to the highly complex nature of the content of the relational data files, associated research papers for the dataset were explored to gain a good background understanding of the data and what it represents.

3. Model Development

The Sci-Kit Learn (v0.23.1) Python package (referred to as sklearn) was used to implement each of the following algorithms. Parameters discussed are referenced with the names given in Python.

3.1 **Learning algorithm 1 -** Decision Tree

Decision trees learn rules from training data to perform classifications on unseen data. The algorithm can be represented as a tree structure where the root node represents the whole training dataset. This node is divided into subsets of data by grouping data together based on a rule. The rules are determined by calculating the entropy of each attribute of the data and selecting the smallest value. When splitting the data into further sub-nodes and creating new rules, attributes used in previous rules are disregarded from the calculation. This splitting will occur until the leaf nodes are met and thus there are no more attributes left to use for decisions. Using a full decision tree (i.e. from the root to the leaf nodes) increases the chance of overfitting the given data which can result in poor performance on unseen data. Pruning the decision tree can reduce the risk of overfitting by removing certain rules, however, a validation set is required to understand the optimal amount of pruning. Decision trees have many parameters that can improve performance. The parameters used in this project were:

- *ccp_alpha* - parameter that controls the pruning of the decision tree
- *criterion* - options for measuring splits are *entropy* and *Gini index*
- *min_samples_leafs* - minimum number of samples required to be a leaf
- *min_samples_split* - minimum number of samples required to split an internal node

Decision trees were chosen due to their methodological rules which can allow for a thorough understanding and conclusion of classification problems. Determining the rules is computed automatically and ensures that less important features do not hinder the classifications. They can work with numerical and categorical data. The algorithm is not heavily sensitive to data that is not scaled or normalised and to data that contains a proportion of missing values.

3.2 **Learning algorithm 2 -** Neural Networks **(**Multi-Layer Perceptron (MLP) and Perceptron)

A Perceptron learning algorithm was first developed by Rosenblatt in 1958 and performs well on data with linear decision boundaries. Weights are assigned to input features and an activation function is used to generate an output. The Perceptron can be extended for non-linear problems into a Multi-Layer Perceptron (MLP) by adding multiple layers of Perceptrons. Both learning algorithms learn the weight values to be able to classify new data. Unfortunately, it can be difficult to understand the results of these learners since the weight and biases implemented happen inside the algorithm, making this a black-box algorithm.

Perceptron:

- *eta0* – the constant that updates are multiplied by
- *max_iter* - number of epochs (i.e. number of times the data passes through the network)

MLP:

- *solver* - changes how the weights are applied. Sigmoid Gradient Descent (*sgd*), uses the gradient of the loss function to apply weights. The Adam solver (*adam*) is similar but the weights are updated while learning. LBFGS (*lbfgs*) can converge faster, performing well on small datasets
- *hidden_layer_sizes* - number of layers in between the input and output layers
- *max_iter* - number of epochs (see Perceptron)
- *learning_rate* - learning rate schedule for weight updates
- *momentum* - momentum for gradient descent update. Only used when *solver* = '*sgd*'
- *nesterovs_momentum* - whether to use Nesterov's momentum used when solver = '*sgd*'
- *learning_rate_init* - the initial learning rate used
- *activation* - activation function choice

3.3 **Learning algorithm 3 -** Bayesian Learning

Naïve Bayes is a classification algorithm that aims to classify the conditional probability of events belonging to a class using Bayes' theorem. A Gaussian Naïve Bayes model is assumed to follow a normal distribution and the number of occurrences is assumed to be independent. The algorithm does not require a large dataset to perform well and is not sensitive to noise in datasets since it assumes all variables are independent of each other. This assumption could increase biases but also makes it less likely to overfit. Naïve Bayes has two parameters: prior and var_smoothing but as there was no prior knowledge for our datasets, var_smoothing was the only parameter explored.

- *var_smoothing* - stability calculation to smooth the curve and account for more samples that are further away from the distribution mean

3.4 **Learning algorithm 4** - Inductive Logic Programming (ILP) and Relational Learning

Similar to the decision tree, rules inferred by ILP are easily understood and interpretable, allowing for concrete conclusions to be made. ILP can be applied to discrete and continuous variables, however, the data has to be prepared and structured differently from the other algorithms. The data must be written using Prolog, a declarative logic programming language.

A Learning **E**ngine for **P**roposing **H**ypotheses (**Aleph**) is an example of an ILP system and can be used to generate hypothesis rule sets for data, developed by Ashwin Srinivasan. For this coursework, it was run using SWI-Prolog but was originally designed for use with the YAP Prolog Compiler. Aleph requires mode declarations, which declare the mode of call for predicates. The predicate *modem* defines those in the literal for a hypothesised clause, with *modeh* used to define the head of a hypothesised clause. Determinations are required to define the predicates that can be used to construct a hypothesis.

There are settings that can be used for the Aleph ILP system, typically defined at the top of the file. For this coursework, settings for Aleph included in a *'aleph.b'* background file provided by Dr Tamaddoni Nezhad were used. The *noise* was set to 30% and setting *i*, the upper bound of layers on new variables was set to 3. The value for noise used was deemed appropriate with findings from [4]. Prolog flags for single variable warnings were set using the *set_prolog_flag* environment control predicate.

3.5 A Note on Grid Search and Randomised Search

Grid search provides a methodical approach to hyperparameter tuning, however has a high time and space complexity. It defines the parameter search space as a grid and evaluates every combination. An alternative method, random search, analyses random combinations of possible hyperparameters in the search space. In general, random search may be preferred as there are fewer searched combinations and the optimal combination is more likely to be found quickly due to the random search pattern. As grid search is exhaustive it was used for all learning algorithms, however, due to the high number of parameters for MLP, random search was used.

4. Model evaluation / Experiments

4.1 Experiment 1 - Rain Dataset

4.1.1 Null Hypothesis I

The null hypothesis for the rain dataset was that the dataset would be best classified using the MLP learning algorithm since the dataset is large and neural networks perform well on large datasets.

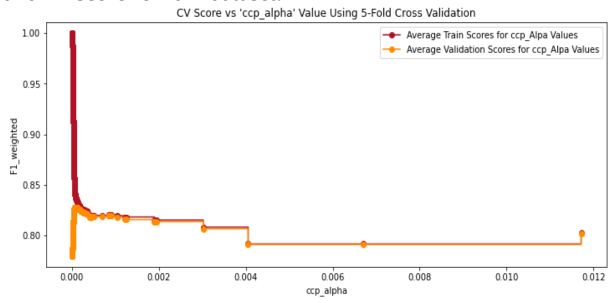
4.1.2 Material & Methods I

After pre-processing, the rain dataset was split into train, validation and test (60:20:20). The test has been kept separate throughout to ensure it was unseen during training and tuning. The train/validation subset was used to train the models and tune the hyperparameters using 5-fold cross validation. Using the default parameters in sklearn, a decision tree was learnt using the training set and an F1 score was found to be 99.99% whereas the validation set gave 77.84%. This indicated that the decision tree was overfitting the data. As discussed in §3, pruning the decision tree can reduce overfitting and the parameter responsible is called ccp_alpha (in sklearn). Applying a minimum cost-complexity pruning algorithm to the decision tree creates an array of ccp_alpha values. Each value corresponds to the number of nodes removed. By learning a decision tree for each of the ccp_alpha values and then calculating the F1 score when classifying the train and validation sets the optimal ccp_alpha value can be found. The optimal value occurs when the maximum F1 score is reached for the validation set and just before it drops. Note that this generally occurs when the accuracy for the training set decreases as increasing the value of ccp_alpha tends to decrease overfitting and can be seen in Fig 2. Stratified cross validation was implemented to ensure the best ccp_alpha was given.

The optimal value was found to be 0.0001367 giving a validation F1 score of 82.4%. The remaining parameters such as the criterion, minimum samples leaf and minimum samples split were tuned with a grid search. As ccp_alpha values are continuous, a discrete set of values were analysed. Optimal values were *entropy* for criterion, *5* for min_samples_leaf and *2* min_samples_split.

Neural networks were explored using the rain dataset. Both Perceptrons and MLP were examined. Using grid search to tune hyperparameters for the Perceptron classifier, values [1, 10, 100, 1000, 10000] were explored for '*max_iter*' and values [0.00001, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 1.0] were explored for *eta0*. Optimal values were found to be 10 and 0.1 respectively, yielding an F1 score of 75.8%.

Figure 2. ccp_alpha performance using 5-fold cross validation and F1 score for rain dataset.



Grid search found the model performed best with *adam* for solver, *False* for nesterovs_momentum, *0.9* for momentum, *300* for max_iter, *0.001* for learning_rate_init, *constant* for learning_rate, *(50,)* for hidden_layers and *logistic* for activation.

Since the rain dataset was normalised and scaled, it could be assumed that the data followed a normal distribution and a Gaussian Naïve Bayes algorithm can be used for classification. The algorithm was trained using the preprocessed data without any hyperparameter tuning, resulting in an F1 score of 0.807 on the test data. When grid search was used to optimise the hyperparameters, the optimal *var_smoothing* value was 0.4329. As a Gaussian Naïve Bayes model gives more weights to the values closer to the distribution mean, this allows the model to add more weights to a higher variance of values accounting for values further away from the mean than before.

4.1.3 Results & Discussion I

Table 1. Algorithm performance for rain dataset.

Algorithms	AUC	F1-Score
Decision Tree	0.847	0.824±0.003
Perceptron	0.811	0.78±0.03
MLP	0.872	0.84±0.002
Naïve Bayes	0.821	0.804±0.002

As shown in Fig 3, the Perceptron has the lowest F1 score and the highest variance. As shown in Fig 4, all the models have a similar AUC; however, the MLP model has the largest area meaning it was able to classify more true positive values per false positive values than any other model.

Figure 3. F1 score comparison for algorithms for rain dataset.

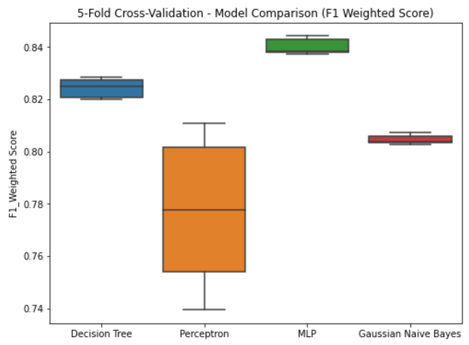
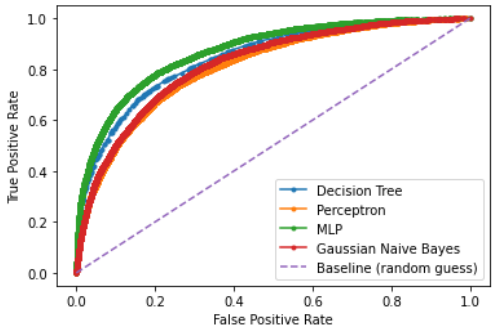


Figure 4. ROC and AUC comparison for rain dataset.



4.2 Experiment 2 - Wine Dataset

4.2.1 Null Hypothesis II

Naïve Bayes was expected to be the most suitable algorithm for the wine dataset as it classifies multiclass problems well. It was expected that due to the high dimensionality, decision trees may perform poorly due to overfitting. MLP was not expected to produce good results as the training data was not large.

4.2.2 Material & Methods II

Following the same methodology as in §4.1.2, hyperparameters for the selected algorithms were tuned. The optimal parameters for the decision tree were *0.000271* for ccp_alpha, *entropy* for criterion, and 1 and 2 for *min_samples_leaf* and *min_samples_split* respectively. The ccp_alpha value was much smaller than the one for the rain dataset, meaning more nodes were pruned. For the Naïve Bayes, tuning showed that the optimal '*var_smoothing*' parameter is 0.0433. This value is smaller than that found for the rain dataset and so less weight is added to the distribution's variance. For the Perceptron the best parameters were: *eta0*=0.0001, *max_iter*=1. The optimal parameters for the MLP were *adam* for solver, *True* for nesterovs_momentum, *0.9* for momentum, *1000* for max_iter, *0.01* for learning_rate_init, *invscaling* for learning_rate *(50,50)* for hidden_layers and *logistic* for activation.

4.2.3 Results & Discussion II

As in Table 2, the best classifier for the dataset was MLP, followed by the decision tree as these algorithms produce the highest F1 and AUC scores. These are the best in classifying wine qualities, despite the MLP misclassifying 479 test examples. The Perceptron was the worst classifier (worse than chance) as there were too many false positives and false negatives. Naïve Bayes also performed poorly, which may be as this classifier assumes the independence of inputs (despite some correlation between the features existing). These results support the null hypothesis.

Table 2. Algorithm performance for wine dataset.

Algorithms	AUC	F1-Score
Decision Tree	0.673	0.577±0.009
Perceptron	0.510	0.39±0.03
MLP	0.744	0.60±0.02
Naïve Bayes	0.565	0.48±0.01

Figure 5. F1 score comparison for algorithms for wine dataset.

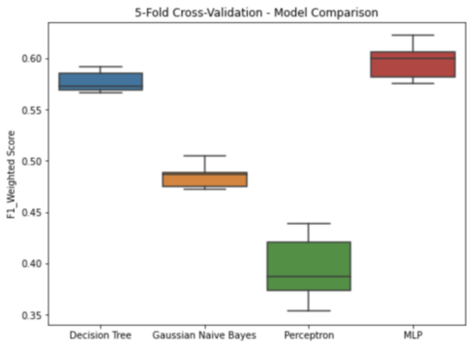
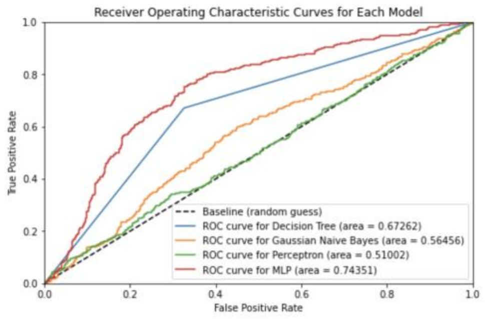


Figure 6. ROC and AUC comparison for wine dataset.



4.3 Experiment 3 - Carcinogenesis Dataset

4.3.1 Null Hypothesis III

The first null hypothesis for the carcinogenesis dataset was that increasing the amount of background information provided to the relational learning algorithm (ILP) would not lead to an increase in classifier performance.

4.3.2 Material & Methods III

To be able to determine if the addition of background information leads to an increase in classification performance, five levels of background knowledge were tested (B0-B4). B0 was the simplest, including information on the atomic structure only; B1 built on this with bond information; B2 and B3 added further information (such as ring information). B4 built on B3, adding rules for mutagenicity found in a research paper using an ILP system [5]. Mutagenicity is the property of a chemical that can induce change in cells or organisms.

Five files were built for each of the levels of background knowledge, with appropriate changes to the '*modeb'* predicates for the system. Once these files had been made, a script from Dany Varghese [6] was adapted to be able to utilise the already pre-split fold data provided in the dataset folder. 10-fold cross validation was then performed using the same folds on each of the 5 background knowledge files using the Aleph ILP system. Accuracy was used as an appropriate metric to use due to the target class balance in the dataset.

4.3.3 Results & Discussion III

From Table 3, there is some evidence to reject the null hypothesis as the 10-fold cross-validation average accuracies can be seen to steadily increase from B0-B2. The average accuracy dips from B2 to B3 however the standard deviation of results also drops, suggesting the model is better at generalising. Performance cannot be said to be improved with the addition of the mutagenicity rules (B4), as the average accuracy falls. This suggests that the addition of mutagenicity rules to the background information does not improve the ability of the ILP system at classification. This decrease is very small, with accuracy still over 93%. Boxplots were plotted so results could be easily compared (see Fig 7(a)).

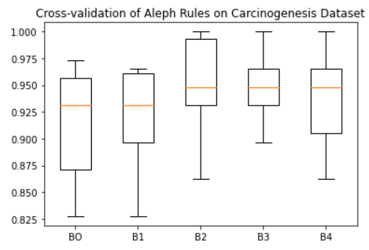
Table 3. Algorithm performance on carcinogenesis dataset with differing amounts of background information.

Algorithm	Background	Accuracy (%)
Aleph	B0	91.45 ± 0.05
	B1	91.87 ± 0.04
	B2	95.25 ± 0.04
	B3	94.56 ± 0.03
	B4	93.87 ±0.04

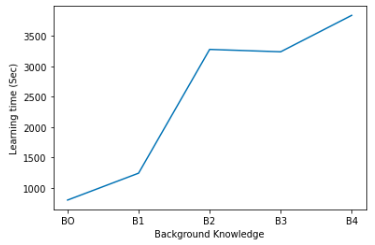
Adding background information increased the time taken by the ILP system, suggesting that adding background knowledge may be more computationally expensive (see in Fig 7(b)).

Overall, it is clear that the addition of more background information does improve the performance of Aleph as there is a general upwards trend in classifier performance as more background knowledge is added. Thus, there are sufficient grounds to reject the null hypothesis. This matches the findings found in other research, supporting this alternate hypothesis [5].

Figure 7. (a) Aleph ILP accuracy results for differing background information. (b) Aleph ILP learning times for differing background information.



(a)



(b)

4.3.4 Null Hypothesis IV

The second null hypothesis for the carcinogenesis dataset was that other non-relational classification algorithms would perform poorly when classifying by only utilising the numerical features of the dataset.

4.3.5 Material & Methods IV

To test the hypothesis, a file containing information from the dataset usable for other algorithms needed to be created. The contents of the *'ind_nos.pl'* file contained numerical count information for Ashby indicators, which are used in toxicology [7]. The Prolog predicates for the Ashby indicator and count were transformed into a CSV file using code that could be called using SWI-Prolog which was adapted from code by Dany Varghese [6]. The Ashby indicator names were the feature names, with the associated chemical in the row.

One of the key issues when converting the data into CSV format was that there were a large number of missing values in each field. For example, there were indicator values for 204 different chemicals but only one instance of the di260 Ashby indicator, resulting in 203 missing values in that column. The sklearn algorithms used in this report do not deal well with missing values and there are a plethora of ways in which missing values can be dealt with, which can affect learning algorithm performance.

As the Ashby indicator values were numerical counts, the obvious choice was to insert a null/zero in place of the missing value. Another choice was to impute values into the empty/missing values. Two popular sklearn methods of imputation are the simple or the iterative imputer [8].

The 'simple' imputer is a basic way for replacing any missing values and was used to replace missing values for the mean (a common practice in machine learning). This is said to be a univariate method as each imputed value only uses information from the column it lies within. On the other hand, the 'iterative' imputer is a more sophisticated approach. It is multivariate as it models the missing values for each feature as a function of the other features in a round-robin fashion. It is an iterative process and as the CSV dataset was small, the maximum number of iterations was set to be 100 for this experimentation.

As additional experimentation, highly sparse columns (with less than 10 non-missing values) were dropped from both the null replacement and the iterative imputed datasets to see the impact on results. The five sets of data were fed into learning algorithms with optimised hyperparameters with the results tested using 10-fold cross validation.

4.3.6 RESULTS & DISCUSSION IV

From the results, it was found that using Ashby indicator counts was not nearly as effective as the Aleph ILP system at classification, however, the traditional algorithms were still quite effective, boasting a better-than-chance accuracy. In Table 4 it can be seen how missing values are processed may suit classifiers differently.

It is shown from the accuracies of the decision tree and Perceptron that univariate imputation leads to a more effective classifier than multivariate imputation for those learners. MLP may misinterpret null values, leading to a poor classifier preferring non-null imputed values. Furthermore, Naïve Bayes struggles with simple imputed values and performs more effectively when the iterative imputer or null values are used.

Table 4. Algorithm performance on carcinogenesis dataset with differing amounts of background information

ALGORITHM	ACCURACY (NULL)	ACCURACY (SIMPLE)	ACCURACY (ITERATIVE)
DECISION TREE	58.55%	62.00%	55.57%
PERCEPTRON	51.40%	54.02%	46.98%
MLP	47.29%	58.55%	56.10%
NAÏVE BAYES	58.45%	41.90%	59.98%

Table 5. Algorithm performance on Carcinogenesis dataset when dropping features with less than10 non-missing values

ALGORITHM	ACCURACY (NULL)	ACCURACY (ITERATIVE)
DECISION TREE	65.52%	60.55%
PERCEPTRON	54.60%	52.12%
MLP	64.47%	58.95%
NAÏVE BAYES	60.98%	60.48%

When comparing the results from Table 5 with Table 4, it is clear that dropping the sparse columns lead to an increase in classifier performance across all algorithms. This was particularly the case for the MLP classifier, which saw a strong increase in accuracy for null-imputed data. This demonstrates that such information may have a high bias, which may negatively affect classification performance.

Furthermore, the null-imputed data with column removal was the most accurate method of processing the data for the traditional sklearn learning algorithms. This can be seen by the higher accuracy scores for all learning algorithms in Table 5.

5. Discussion of the results, interpretation and critical assessment

Based on F1 and AUC, MLP was the best model for both the rain and wine datasets, followed by the decision tree algorithm, with Perceptron performing poorly. Differences between the datasets led to all classifiers performing better on the rain data. This is mainly attributed to the quality of data and the data problem. Performing a multiclass classification on a highly unbalanced dataset proved difficult for all algorithms.

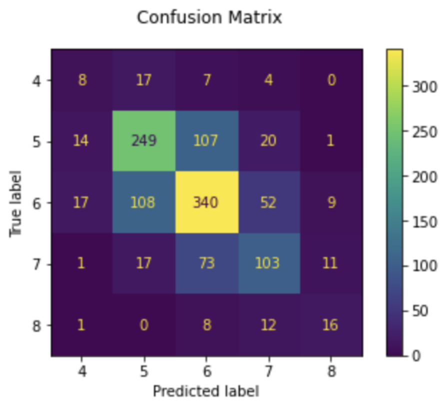
All classifiers had F1 scores below 60% when used on the wine dataset. Fig 8 shows an example of a confusion matrix, demonstrating that even the best performing algorithm, the MLP, misclassified many instances. Such performance could also be attributed to wine quality being subjective, so, likely, there were not clear separable labels (noise). The rain dataset presented a simpler binary classification problem and had sufficient data, resulting in all algorithms performing well. In both cases, the lack of knowledge about the prior probability of classes as well as the violation of the conditional independence assumption limited the usefulness of the Naïve Bayes classifier. Hence, its performance was mediocre in both experiments.

Experiment III highlighted shortcomings of non-relational algorithms as the lack of records resulted in relatively poor performance (no setup achieved more than 66% accuracy). When the data is structured in a form appropriate for relational learning and background knowledge is provided, as seen in Table 3, the ILP learning algorithm significantly outperformed the other algorithms, giving higher than 90% accuracy.

Despite the MLP's good performance, it is inherently unexplainable. In contrast, decision trees and ILP systems are highly explainable. Their output is easy to understand as a set of rules is produced, so they are more useful for inference related tasks and are better suited for decision making. This also makes them more interesting to apply as their decisions are comprehensible and can be traced back. Nevertheless, decision trees may grow exponentially for large data, as demonstrated by Experiments I and II. Such trees can be complex and hard to navigate. Moreover, decision trees can be unstable and significantly affected by small variations in data. They are built by taking locally optimal decisions at each node, which means there is no guarantee of a globally optimal solution, finding which is

NP-complete. Plus, highly unbalanced datasets, such as the wine dataset, can result in a biased decision tree.

Figure 8. Confusion matrix for the MLP - wine dataset



6. Conclusions

Overall, the project showed that certain models perform significantly better or worse depending on the data they are applied to. It was shown that MLPs perform well on large datasets in binary and multiclass classification problems. Decision trees should be preferred when explainability is important. Hence, MLPs performance was the best when used with the rain and wine dataset.

Some models are too simple for classifying data in such settings. Perceptron cannot learn non-linear functions and Naïve Bayes's conditional independence assumption does not always hold, which could be addressed by removing more correlated features. These classifiers showed the worst performance.

A conclusion made from the relational dataset was that the Aleph ILP learning algorithm is a far better classifier than traditional learning algorithms for the highly complex relational chemistry data provided. It was demonstrated that the addition of background information has an advantageous effect on classification, despite increasing average learning time. Comparisons were made between different methods of imputation for the missing values for Ashby indicator count data, finding that different algorithms prefer different imputation methods. Learning algorithms were found to perform better when highly sparse features were dropped. Overall, the Aleph ILP learner was an excellent classifier, achieving over a 95% accuracy on the carcinogenesis dataset and providing a strong resolution to the data problem.

In future work, a resampling technique to solve the class imbalances could be tested, potentially improving classifier accuracy. Limiting the number of MLP layers and extracting symbolic rules to improve transparency and explainability could also be an option. Moreover, many of the limitations associated with decision trees could be addressed with the implementation of a random forest ensemble method, improving the results. Lastly, one-hot encoding could be explored for the rain dataset (instead of label encoding), adding order to the variables and potentially improving classifier performance.

Contributions

The workload for the project was split into three. Danil and Will worked primarily on the wine dataset; Emily, Jenna and Jamie worked primarily on the rain dataset and Will worked on the ILP dataset. All group members attended all meetings, wrote together to formulate the report and collaborated effectively.

Acknowledgments

The group would like to thank Dr Tamaddoni Nezhad for hi invaluable time and help as we were completing this project. We would also like to thank Dany Varghese for providing us with his Aleph Python implementation code and help during lab sessions.

References

[1] Kaggle, Rain in Australia. Accessed March 2021. https://www.kaggle.com/jsphyg/weather-dataset-rattle-package

[2] Kaggle. Wine Quality Dataset. Accessed March 2021. https://archive.ics.uci.edu/ml/datasets/Wine+Quality

[3] GitHub. Carcinogenesis Dataset. Accessed May 2021. https://github.com/JoseCSantos/GILPS/tree/master/datasets/carcinogenesis

[4] A. Srinivasan, R.D. King, S.H. Muggleton, M.J.E. Sternberg. Carcinogenesis predictions using ILP. Lavrač N., Džeroski S. (eds) Inductive Logic Programming. ILP 1997. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 1297. Springer, Berlin, Heidelberg.

[5] A. Srinivasan, R.D. King, S.H. Muggleton. The role of background knowledge: using a problem from chemistry to examine the performance of an ILP algorithm. 1996. [https://citeseerx.ist.psu.edu/viewdoc/down*load?doi=10.1.1.23.8422&rep=rep1&type=pdf]*

[6] Dany Varghese, COMM055 Machine Learning and Data Mining Lab Sheet 7, 2021

[7] Claire Julia Kennedy. *Strongly Typed Evolutionary Programming*. University of Bristol. Page 123. 1999. [https://research-information.bris.ac.uk/en/studentTheses/strongly-typed-evolutionary-programming-2]

[8] Sci-Kit Learn documentation, accessed May 2021. http://scikit-learn.org/stable/modules/impute.html