CS504 Spring 2021 Project 3

Data Exploration:

1. Types of attributes

Dataset a.arff

x - Numerical Continuous (real valued attribute)
y - Numerical Continuous (real valued attribute)
class – Categorical Nominal

Dataset b.arff

x - Numerical Continuous (real valued attribute)
y - Numerical Continuous (real valued attribute)
class – Categorical Nominal

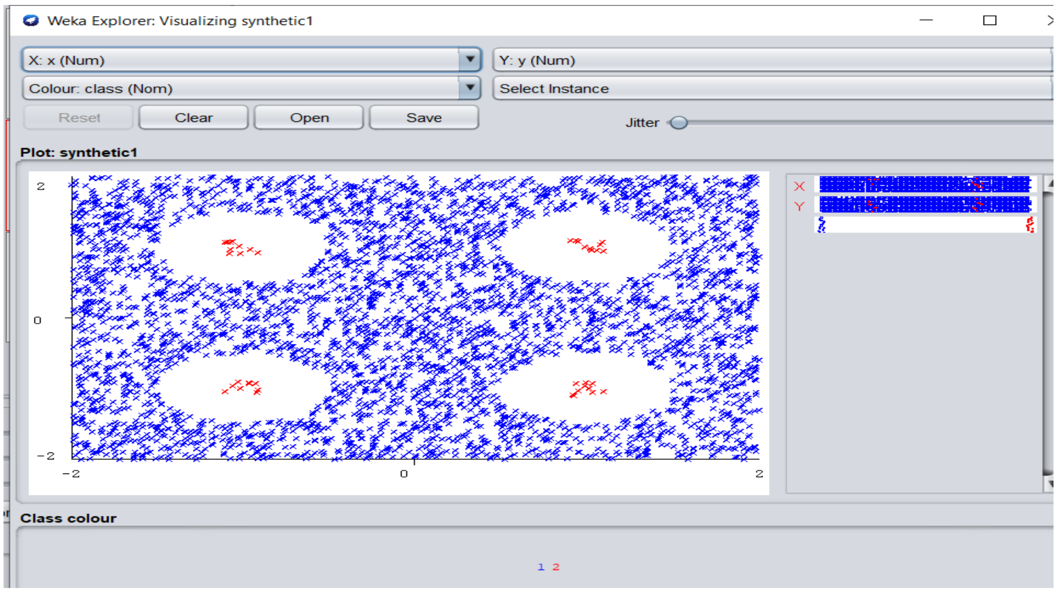
Dataset c.arff

a0001 - Categorical Nominal
a0002 - Categorical Nominal
a0003 - Categorical Nominal
a0004 - Categorical Nominal
class - Categorical Nominal
All the attributes from a0005 to a0099 comes under Numerical Continuous (real valued attributes)

2. Class distribution

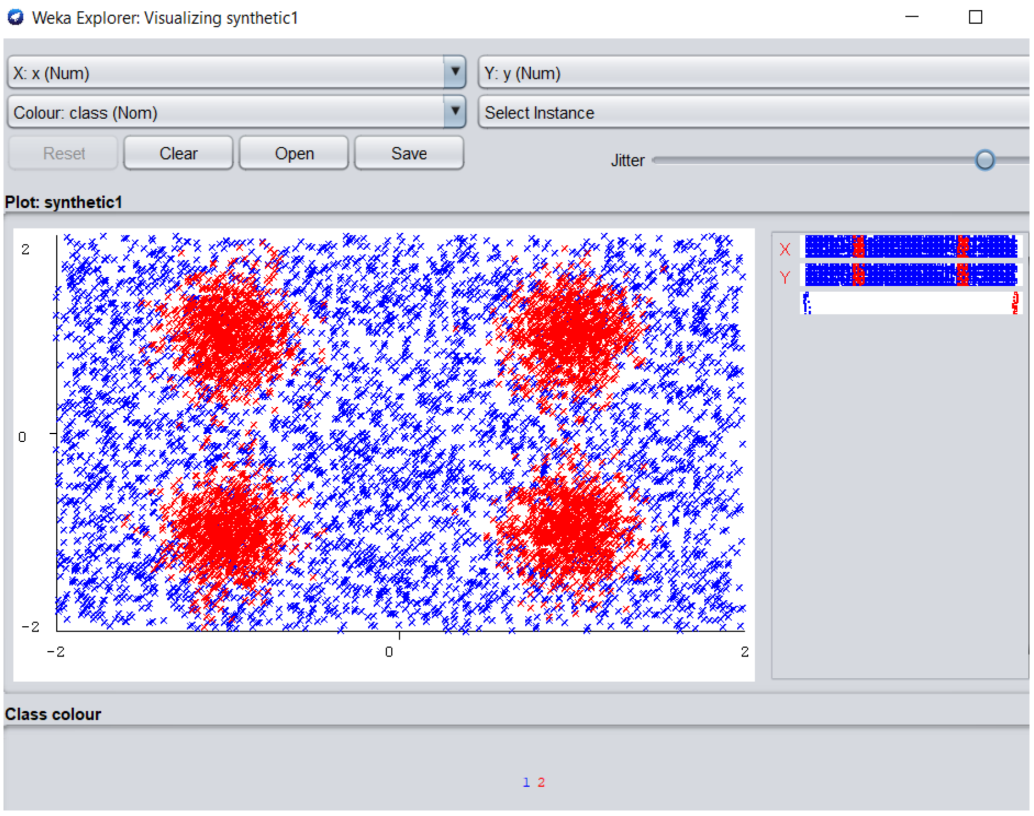
Dataset a.arff - 99% for class with class value '1' and 1% for the class with class value '2'. Data set A is imbalanced dataset.
Dataset b.arff - 50% for each of 2 classes (1 and 2). It is a balanced dataset.
Dataset c.arff - 50% for each of 2 classes (1 and 2). It is a balanced dataset.

3. Any special structure that you might observe, if any : Dataset a.arff - Cluster formation



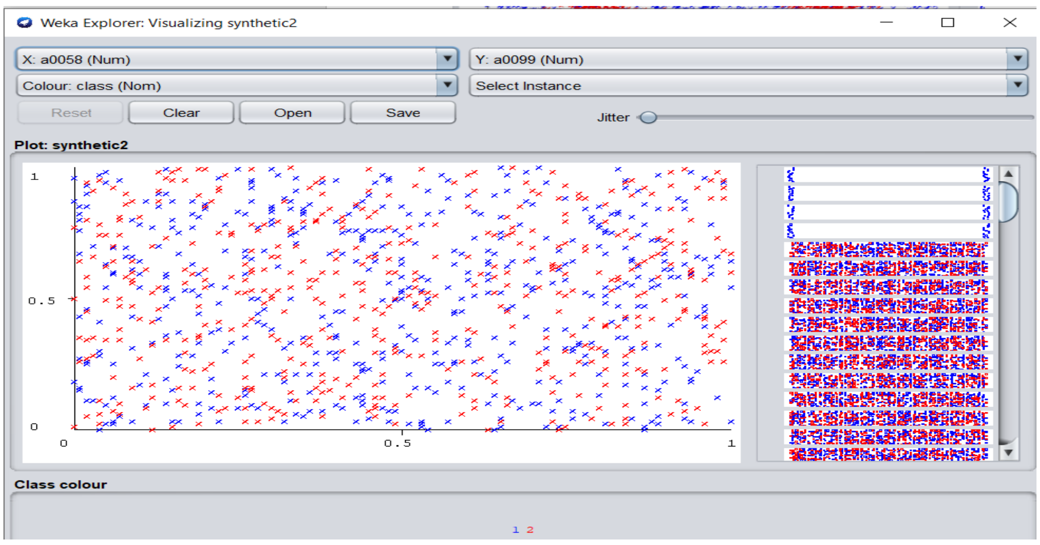
There is a cluster formation for class value ‘2’ when (x, y) = (-1, -1), (1, -1), (-1, 1), (1, 1) [these values of x and y are not exact, they may change on increasing and decreasing the jitter in the above figure] and the remaining area is occupied by the class value ‘1’ as shown in the above visualization. In this dataset the instances for class value 2 are very low in number.

Dataset b.arff : Cluster formation



It is clearly visible that when x and y values are plotted on x axis and y axis along with class values, clusters are formed. In this data set there are equal instances of class values 1 and 2.

Dataset c.arff :



There is no special structure for this dataset when visualized. The data is just scattered all around.

Experiment 1:

In dataset C, data is equally distributed between two classes 1 and 2.

1. Trees->DecisionStump

Correctly Classified Instances 520 52 %

Root mean squared error 0.5004

2. Trees->J48

Correctly Classified Instances 910 91 %

Root mean squared error 0.2923

3. Trees->J48 Unpruned

Correctly Classified Instances 696 69.6 %

Root mean squared error 0.5417

CLASSIFIER	CLASSIFICATION ACCURACY	ROOT MEAN SQUARED ERROR (RMSE)
DecisionStump	52%	0.5004
J48 pruned	91%	0.2923
J48 Unpruned	69.6%	0.5417

For DecisionStump, briefly explain the technique and list the attribute that was used to make the decision. Compare the results of J48(pruned/unpruned) and explain why pruned has better performance

DecisionStump:

1. Decision Stump is a type of Machine Learning Model having Single level Decision Tree i.e. a decision tree with one internal or top node which is connected to its leaf nodes. (otherwise known as terminal nodes).
2. Decision Stump makes predictions based on values of single independent variable or input feature
3. This Machine Learning model selects that one single input feature that has the least classification error.
4. Missing values are considered as another category. This model works with both nominal and binary independent variables.
5. For nominal independent variables, stump which contains leaf for each input feature value is built.

6. Feature Splitting learning problem can be viewed as problem of learning Decision stump as it is a one level decision tree.

Reference: https://www.youtube.com/watch?v=mSmbKt3qccc

The attribute that was used to make the decision: a0079

This input feature is chosen by the DecisionStump tree model to make prediction because after taking into account all the input features and calculating the classification error for each of those, a0079 input feature has the least classification error. So a0079 is used to make a decision.

J48 Pruned and Unpruned Comparison:

	J48 Pruned	J48 Unpruned
Number of Leaves	4	41
Size of the tree	7	81
Correctly Classified Instances	91%	69.6%
Incorrectly Classified Instances	9%	30.4%
Kappa statistic	0.82	0.392
Mean absolute error	0.0903	0.305
Root mean squared error	0.2923	0.5417
Relative absolute error	18.0574%	60.9914%
Root relative squared error	58.452%	108.331%
Time taken to build model	0.25 sec	0.09 sec
TP Rate (Weighted Average)	0.910	0.696
FP Rate(Weighted Average)	0.090	0.304
Precision (Weighted Average)	0.910	0.696
Recall (Weighted Average)	0.910	0.696
F – Measure (Weighted Average)	0.910	0.696
MCC (Weighted Average)	0.820	0.392
ROC Area (Weighted Average)	0.938	0.733
PRC Area (Weighted Average)	0.914	0.683

Pruning is nothing but removing the nodes or branches that do not have any affect on the performance. It can also be seen as removal of decisions that do not have much benefit. The reason why pruning is done is it helps to understand the tree in an easier way and we can reduce the risk of overfitting the training data. ***Unpruning*** on the other side is opposite to what pruning does i.e., we allow the tree to grow till the end and this may over fit the training data.

1. As there are no removal of nodes in the Unpruned model, the tree size is generally larger (81) and the terminal nodes (41) are more in count compared to the Pruned model.

2. Pruned model is able to classify 91% of the instances correctly which is way larger than the 69.6% by unpruned model. So Pruned model is more accurate compared to unpruned.
3. There is almost perfect standardized measure of agreement i.e. Kappa Statistic (0.82) between two class values 1 and 2 in the pruned model. There is a fair agreement (0.392) between two class values 1 and 2 in the unpruned model.
4. The errors namely Mean absolute error, Root mean squared error, Relative absolute error, Root relative squared error for Pruned model are very less compared to the unpruned model which can be seen in the above table. This say that pruned model is able to make predictions or decisions with much less error compared to unpruned.
5. Time taken to build the pruned model (0.25 sec) is more compared to the unpruned with just 0.09 sec as pruned model takes time in removing the nodes without affecting the performance.
6. Models here says so and so instances belong to class 1 and class 2. What percent of the time was it correct can be explained from Precision. The weighted average precision for pruned model is more (0.910) compared to unpruned with 0.696 weighted average precision.
7. If some instances actually belong to class 1 or class 2, what percent of the time did the model predict that they belong to class 1 or class 2 can be explained by Recall. The weighted average Recall for pruned model is more (0.910) compared to unpruned with 0.696 weighted Recall.
8. The accuracy of the pruned model can be seen through True Positive Rate and False Positive rate. The TP Rate is high for pruned model (0.910) i.e. pruned model correctly predicts the actual positive values at a high rate compared to Unpruned. For Unpruned model 30.4 is incorrectly predicted positive percentage whereas only 9 is the incorrectly predicted positive percentage by the pruned model.
9. We can see there is a perfect performance by the pruned classifier which can be explained through F-Measure. F measure is generally the harmonic mean of precision and recall. F measure is 0.910 for pruned model and 0.696 for unpruned. From the data we can say that Pruned model has better performance than the Unpruned model.
10. Matthews correlation coefficient (MCC) or phi coefficient is high for pruned compared to unpruned. From this we can say that the quality of binary classification is high for Pruned models compared to unpruned models
11. ROC curve is nothing but the Receiver Operating characteristics Curve. This curve says how a particular classifier is performing in general. The ROC area for pruned model is 0.938 which indicates that prune model is performing better compared to the unpruned model which has an ROC area of 0.733. Also, Precision Recall Curve area for the pruned model is higher compared to the Unpruned model.

Why do Pruning has better performance?

From the above comparisons it is clear that Pruning has a better performance than Unpruning. To conclude, the evaluation metrics namely Receiver Operating characteristics Curve area, F - measure, Classification Accuracy are higher for pruned models compared to unpruned. Root Mean Squared Error is very low for pruned models. These all metrics indicate that Pruned models have better performance than the Unpruned.

Reference:

1. Class Notes
2. https://stackoverflow.com/questions/11585715/what-is-pruned-and-unpruned-tree-in-weka

Experiment 2:

CLASSIFIER	**F – measure (weighted avg)** **Dataset a**	**F – measure (weighted avg)** **Dataset b**
J48 pruned	**0.994**	**1.000**
NaiveBayes	**?**	**0.744**
IBk (k=1)	**1.000**	**1.000**
IBk (k=21)	**?**	**1.000**

Explanation:

Data set A is an imbalanced dataset with 99% of the data in class 1 (4023 instances) and just 1 %(40 instances) of the data in class 2 whereas dataset B is a balanced dataset with approximately equal instances for both class values 1 and 2. F measure is generally the harmonic mean of precision and recall. Perfect performance of the classifier can be explained by the F measure model estimate.

Compare Performance of J48 (pruned) on Datasets A and B:

F measure is 0.994 for dataset A and 1.000 for dataset B with J48 (pruned) classifier. This data explains that Data set B has a better performance than dataset A. Also, there is equal class distribution in dataset B. Dataset A is an imbalanced dataset and f-measure is 0.994, this may be misleading as model may not detect any class 2 samples. So dataset B has better performance than A.

Compare Performance of NaiveBayes on Datasets A and B:

F measure is null value for dataset A and 0.744 for dataset B with NaiveBayes classifier. The data explains that Data set B has a better performance than dataset A. F measure is calculated based on Precision and Recall. As the weighted average precision is null, F score is null for data set A.

Compare Performance of IBk (k=1)[instance based k] on Datasets A and B:

F measure is 1.000 for dataset A and 1.000 for dataset B with IBk (k=1) classifier. The data explains that 1.000 is the perfect F measure, but with hyper parameter k =1, the data is over fitted and this model may not work well with the unseen data and can't be generalized. So, though there is a perfect performance by IBk (k=1) classifier, overfitting of data will happen (as it is sensitive to noise points, complex model) with both the data sets A and B and this F measure may not make sense.

Compare Performance of IBk (k=21) on Datasets A and B:

F measure is null for dataset A and 1.000 for dataset B with IBk (k=21) classifier. Here the k value is tuned to 21 and the model performance can be seen to be perfect for the dataset B.

Now, there can be more reasonable decision boundary and this model can be less susceptible to over fitting when k =21. Generally, one of the way to find the value of k is to tune the value of k till we get the lowest error rate based on the test set.

Compare Performance of the 4 classifiers on Dataset A

	J48 (pruned)	**NaiveBayes**	**IBk (k=1)**	**IBk (k=21)**
Correctly Classified Instances (in %)	99.5078 %	99.0155 %	100 %	99.0155 %
Root mean squared error	0.0699	0.0987	0.0003	0.06
Time taken to build model (in sec)	0.01	0.01	0	0
Precision	0.995	?	1.000	?
Recall	0.995	0.990	1.000	0.990
F - measure	0.994	?	1.000	?
ROC Area	0.993	0.557	1.000	1.000

Give explanations for your observations (from experiment 2)

1. F measure is 0.994, null, 1.000, null for dataset A with J48 (pruned), NaiveBayes, IBk (k=1), IBK (k=21) classifiers respectively. F – measure of 1 can have perfect performance but it may overfit the data when k = 1 in Instance Based Learning Classifier. So the next better performance model is J48 (pruned).
2. F measure with null values makes no sense as the precision is null and we can't predict any kind of performance from NaiveBayes and IBk (k=21) with this data set.
3. Also, J48 (pruned) and Naïve Bayes classifiers take time (0.01 sec) to build the model, whereas instance based learning do not build any model so the time take is 0 sec.
4. The Root mean squared error is highest for the Naïve Bayes classifier compared to other classifiers used on dataset A.
5. Finally, ROC area is highest for IBk (k=1) but with hyper parameter value 1, this may be a complex model and overfitting can be observed. So we consider the next best ROC area value which is 0.993 for J48 (Pruned).
6. With k =1 IBk classifier overfits the data. So, considering the model estimates for this classifier will not be apt.
7. Overall, J48 (pruned) classfier is best suited for dataset A according to the above data. However, class distribution in dataset A is imbalanced. If this is balanced, may be other classifiers can have better performance.

Experiment 3:

Compare Performance of the 3 classifiers on Dataset C:

CLASSIFIER	**F- measure (Dataset c)**
NaiveBayes	**0.472**
IBk (k=1)	**0.971**
IBk (k=10)	**1.000**

1. F measure is 0.472, 0.971, 1.000 for dataset C with NaiveBayes, IBk (k=1), IBK (k=10) classifiers respectively.
2. If F measure is 0 to 0.3 then the classifier performance is at worst. If 0.4-0.5 a moderate performance, 0.6-0.7 fair performance and 0.8-1.0 best performance.
3. So, Naïve Bayes has a moderate performance with dataset C with F – measure 0.472.
4. IBk with hyper parameter k=1 has best performance too but this can overfit the data.
5. IBk with hyper parameter k = 10 has the best performance among all the 3 classifiers when considering the F- measure model estimate.
6. F – measure is generally calculated from precision and Recall. That is, it is the harmonic mean of Recall and Precision
7. Finally, IBK with k = 10 has F-measure 1.000 which gives best performance as it do not overfit the data compared to when hyper parameter is tuned to 1.

Experiment 3- Comment on the effect of k.

Instance based learning classifier do not build any model. It compares new instances with training instances. K – Nearest neighbour is an example of Instance based learning. K is nothing but the number of nearest neighbours it takes into consideration when making a prediction or decision. Finding K values can be tough and challenging. When K is too large, neighbourhood may include points of other classes. When K is too small, it is susceptible to noise points. Greater the K value, the more the estimates are smoothed out among neighbours. When k is very low like k=1, then the model is complicated and over fitted. When k is very low like k=n, then the model is simple and under fitted. Here n is the number of records in the training example. When k =n, entire dataset is used for every prediction which simply predicts the majority class in the training example. When we consider the k value between 1 and n, then we can get a more reasonable decision boundary. We should tune the k value till we reach the lowest error rate based on the test data. This is one way of finding the k value.

In the dataset C, we have 1000 instances. When k = 1, overfitting can be seen but the f – measure is approximately 1. So, because of noise points we don’t consider k=1. When k =1000 then under fitting can be observed. Here the k value is selected as 10, so this can give us a more reasonable decision boundary. It doesn’t hurt if k – value is between 10 – 15. This is the effect k has on the model estimates in Instance based learning classifier.

Explanation:

From the above comparison it is clear that IBk (k=10) have the best performance compared to IBk (k=1) and Naïve Bayes. We can also use ROC area to better estimate the performance of the classifiers along with F – measure.