**Part 1: Classification with Neural Networks**

1. We are using the Weka application for this assignment to do the analysis for the part 1 which is Classification with Neural Networks. We have used hypothyroid.arff. dataset for our analysis, First of all we are doing the data pre-processing which is done by weka itself by using the filter which is in preprocess. After going in preprocess then by clicking on choose which is in filter and then by selecting Preprocess>filters>unsupervised> attribute and in attribute select ReplacewithMissingValues which replaces all the missing values for nominal and numeric attributes in a dataset with the modes and means from the training data. We then Normalized the numeric attributes by excluding class attributes and by using the filter which is in weka>filters>unsupervised> attribute>Normalize and then by clicking apply all the filters will be applied. There will be 33 inputs and 4 outputs. So in this dataset we are not converting the nominal attributes to binary at first but we will be doing this while classifying which is in classify>choose>weka>classifier>bayes>functions>MultilayerPerceptron by editing the properties of the classifier and setting the nominaltobinary to true.
2. We are using 66% of the dataset into training dataset and then the remaining to testing and validation datasets without any duplication. To generate the necessary training, validation and test data files we split the dataset by using the Resample filter which is in Preprocess>filters>unsupervised> instance and then by selecting resample and then by selecting “noReplacement” to true to avoid duplicates and then also by selecting “sampleSizePercent ” to 66%. After that by clicking on apply and save the file as training.arff. Now for producing testing and validation sets we first do undo so that we get back to original and then by setting “invertSelection” to True since we want to get 34% as testing.arff. When testing set is produced we then used it by using Supplied Test Set for our further analysis. Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Now to stop training a neural network we get to know that by using validation set, so when the error goes down and at a certain time the error starts to go up again this is the time to stop the training to avoid overfitting.
3. All the neurons has some weightage calculated using non-linear function from the previous layer neurons. The initial neurons of a trained neural network has weightage values calculated based on the training values. The trained neural network compares these weightages when applied on test data. To improve the accuracy neural network is trained multiple times (epochs) refining the initial weightages to reduce error using back propagation (Delta rule) based on the training values randomly.
4. Table for No Hidden Layer Used.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Architecture | Parameters | Train MSE | Train Error | Epochs | Test MSE | Test Error |
| 1 | ii-hh-oo  (33-00-4) | Lr=0.3 | 0.02322 | 5.4083 % | 500 | 0.029275 | 6.3183 % |
| 2 | ii-hh-oo  (33-00-4) | Lr=0.3 | 0.023225 | 5.4083 % | 1000 | 0.029275 | 6.3183 % |
| 3 | ii-hh-oo  (33-00-4) | Lr=0.4 | 0.0227 | 5.1962 % | 500 | 0.029894 | 6.4743 % |
| 4 | ii-hh-oo  (33-00-4) | Lr=0.4 | 0.02064 | 4.7985 % | 1000 | 0.029963 | 6.5523 % |
| 5 | ii-hh-oo  (33-00-4) | Lr=0.4 | 0.018714 | 4.3478 % | 2000 | 0.030206 | 6.6303 % |

We can see from the above results that as we are increasing the Learning rate and the epochs the training and testing error are not changing very drastically and also the Train and test MSE have slightly changed. A single layer perceptron have only one layer and can learn linear functions whereas the multilayer perceptron has one or more hidden layers and can also learn non-linear functions. Single layer perceptron can only represent a limited set of functions and has the limitation of not learning efficiently whereas the multilayer perceptron learns very effectively because of its many layers.

1. Assuming one hidden Layer and 5 hidden nodes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Architecture | Parameters | Train MSE | Train Error | Epochs | Test MSE | Test Error | Overfitting |
| 1 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.024273 | 5.7529 % | 500 | 0.027989 | 6.1622 % | 0.4093  NO |
| 2 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.024273 | 5.7529 % | 1000 | 0.027989 | 6.1622 % | 0.4093  NO |
| 3 | ii-hh-oo  (33-01-4) | Lr=0.4 | 0.023470 | 5.8324 % | 500 | 0.027989 | 6.3183 % | 0.4859  NO |
| 4 | ii-hh-oo  (33-01-4) | Lr=0.4 | 0.023470 | 5.8324 % | 1000 | 0.027989 | 6.3183 % | 0.4859  No |
| 5 | ii-hh-oo  (33-01-4) | Lr=0.4 | 0.023470 | 5.8324 % | 2000 | 0.027989 | 6.3183 % | 0.4859  No |

As we can see from the above tables that when 5 hidden nodes are used, on changing the learning rate there are not any major change on the results and also when we changed the epochs we also got the same results. On increasing the learning rate and epochs we can also say that there not any major change in test and train errors and also there is no overfitting as we can see the difference between the test and train errors is very low.

1. For one layer with multiple hidden nodes.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Hidden Nodes | Architecture | Parameters | Train MSE | Train Error | Epochs | Test MSE | Test Error | Overfitting |
| 1 | 4 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.02461 | 5.8324 | 500 | 0.02802 | 6.8643 | 1.03  Yes |
| 2 | 6 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.02347 | 5.5673 | 500 | 0.027688 | 6.3183 | 0.75  No |
| 3 | 8 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.02187 | 5.2492 | 500 | 0.028156 | 6.6303 | 1.39  Yes |
| 4 | 10 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.02067 | 4.6394 | 500 | 0.027855 | 6.3963 | 1.76  Yes |
| 5 | 12 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.02298 | 5.4878 | 500 | 0.02839 | 6.4743 | 0.99  No |
| 6 | 15 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.02390 | 5.6734 | 500 | 0.02798 | 6.5523 | 0.88  No |
| 7 | 20 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.01617 | 3.8971 | 500 | 0.02768 | 6.3183 | 2.42  Yes |
| 8 | 30 | ii-hh-oo  (33-01-4) | Lr=0.3 | 0.02292 | 5.3552 | 500 | 0.027889 | 6.2402 | 0.905  No |

From the above results we can say that the best hidden node is 6.As there is no overfitting. We can also see from the results that node 20 has the overfitting of 2.42.To conclude we can say that for this problem node 6 is better and as the nodes keeps on increasing there exists the overfitting and also more the hidden nodes, more will be the complexity and the problem of overfitting can also arise.

7. For one Hidden Layer with 5 hidden nodes and with different Learning rate and Momentum.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Hidden Nodes | Architecture | Parameters | Train MSE | Train Error | Epochs | Test MSE | Test Error |
| 1 | 5 | ii-hh-oo  (33-01-4) | Lr=0.3  Momentum=0.2 | 0.02427 | 5.7529 | 500 | 0.02798 | 6.1622 |
| 2 | 5 | ii-hh-oo  (33-01-4) | Lr=0.4  Momentum=0.3 | 0.02362 | 5.5143 | 500 | 0.02835 | 6.2402 |
| 3 | 5 | ii-hh-oo  (33-01-4) | Lr=0.6  Momentum=0.5 | 0.02676 | 6.1241 | 500 | 0.02879 | 6.5523 |
| 4 | 5 | ii-hh-oo  (33-01-4) | Lr=0.8  Momentum=0.7 | 0.02805 | 6.2036 | 500 | 0.02917 | 6.5523 |
| 5 | 5 | ii-hh-oo  (33-01-4) | Lr=0.9  Momentum=0.9 | 0.03701 | 7.7147 | 500 | 0.03701 | 7.7147 |

From the above results we can say that as the learning rate and momentum are increased the train and test error keeps on increasing, the model is unable to learn properly and model takes more time to train. The model works better when learning rate is 0.3 and Momentum is 0.2. To conclude we can say that, the model learns well when the learning rate is minimum.

8. Table for Multilayer Perceptron.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Architecture | Parameters | Train MSE | Train Error | Test MSE | Test Error | Train Accuracy | Test Accuracy |
| 1 | ii-hh-oo  (33-00-4) | Lr=0.3 | 0.02322 | 5.4083 | 0.02927 | 6.3183 | 94.5917 | 93.6817 |
| 2 | ii-hh-oo  (33-00-4) | Lr=0.4 | 0.02277 | 5.1962 | 0.02989 | 6.4743 | 94.8038 | 93.5257 |
| 3 | ii-hh-oo  (33-00-4) | Lr=0.6 | 0.02669 | 5.6999 | 0.03041 | 6.4743 | 94.3001 | 93.5257 |

Table for J48 Classifier for C and M parameter.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Run No | Parameters | Train Error | Test Error | Training Accuracy | Testing Accuracy | Overfitting |
| 1 | C=0.25 & M = 2 | 0.1856 | 0.546 | 99.8144 % | 99.454 % | NO |
| 2 | C= 0.25 & M=3 | 0.3181 | 0.546 | 99.6819 % | 99.454 % | NO |
| 3 | C = 0.5 & M= 5 | 0.2916 | 0.546 | 99.7084 % | 99.454 % | NO |

From the above tables we can conclude that the accuracy of J48 Classifier is way better than Multilayer Perceptron. Also J48 is less time consuming that Multilayer Perceptron due to its feed feed-forward artificial neural network. J48 has better performance as seen from the above tables than MultiLayerPerceptron.

9. JavaNNS helps with the error graph which provides the graph visualization. While watching this graph. We can see the error MSE and also we get to know when to stop the model by watching at its validation part. When the validation graph falls down and again it starts rising we can know that stop the training data to avoid overfitting. We can also use analyse function in javaNNS which helps to test and validate the data accuracy and errors which are done by test.res and validate.res files.JavaNNS also doesn’t have the statistical analysis we can only see the graph at once unlike WEKA we can see results in one page itself. There is no option of preprocess in JavaNNS but we need to develop the script to preprocess the data and splitting the training, testing and validation files. Making script in javaNNS takes time. In JavaNNS we can make our own neural network.

WEKA is the best software for calculating the Machine Learning algorithms accuracy and for data mining. In WEKA we can do the pre-processing by the filters present in it. In WEKA we can use filters by which we can divide the datasets into training, testing and validation sets by using its properties as shown earlier. In testing set we can use validation set size from the properties.It also provides the statistical calculations in the same window. In WEKA we only have MSE calculations but there is no error graph like javaNNS. In WEKA we can see the Neural Network by setting GUI to True. Weka is better with its User Interface and also with its fast performance and for this reason we select WEKA to do this assignment.

**Part 2: Numeric Prediction with Neural Networks**

1. We are using the Weka application for this assignment to do the analysis for the part 2 which is Numeric Prediction with Neural Networks. We have used heart-v1.arff. dataset for our analysis, First of all we are doing the data pre-processing which is done by weka itself by using the filter which is in preprocess. After going in preprocess then by clicking on choose which is in filter and then by selecting Preprocess>filters>unsupervised> attribute and in attribute select ReplacewithMissingValues which replaces all the missing values for nominal and numeric attributes in a dataset with the modes and means from the training data. We then Normalized the numeric attributes by using the filter which is in weka>filters>unsupervised> attribute>Normalize and then by excluding class attributes by clicking apply all the filters will be applied. There will be 23 inputs and 1 output which is “chol” attribute. So in this dataset we are not converting the nominal attributes to binary at first but we will be doing this while classifying in classify>choose>weka>classifier>bayes>functions>MultilayerPerceptron by editing the properties of the classifier and setting the nominaltobinary to true.
2. We are using 70% of the dataset into training dataset and then the remaining to testing and validation datasets without any duplication. To generate the necessary training, validation and test data files we split the dataset by using the Resample filter which is in Preprocess>filters>unsupervised> instance and then by selecting resample and then by selecting “noReplacement” to true to avoid duplicates and then also by selecting “sampleSizePercent ” to 70%. After that by clicking on apply and save the file as training.arff. Now for producing testing and validation sets we first do undo so that we get back to original and then by setting “invertSelection” to True since we want to get 40% as testing.arff. When testing set is produced we then used it by using Supplied Test Set for our further analysis. The Mean-absolute error in weka is calculated by :

For every instance in the test set, weka obtains a distribution (for each class label a value from 0 to 1, i.e., 0-100%). This distribution is matched against the expected distribution. For each class label the following is calculated: AbsErrPerLabel = abs (actual - predicted)/No of class labels and then the sum of this errors is done. The mean absolute error is calculated as:

Sum (AbsErrorPerInstance)/number of instances with classlabel. Normalisation is already done to numeric variables to facilitate the calculation as explained earlier.

1. For No Hidden layer used which is single layer perceptron.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Run No | Architecture | Parameters | Train MAE | Epochs | Test MAE |
| 1 | ii-hh-oo  (23-00-1) | Lr=0.3 | 42.5771 | 500 | 42.1177 |
| 2 | ii-hh-oo  (23-00-1) | Lr=0.5 | 26.179 | 1000 | 24.7573 |
| 3 | ii-hh-oo  (23-00-1) | Lr=0.01 | 16.8853 | 2000 | 16.6985 |
| 4 | ii-hh-oo  (23-00-1) | Lr=0.02 | 17.8403 | 3000 | 17.2819 |
| 5 | ii-hh-oo  (23-00-1) | Lr=0.002 | 16.3553 | 5000 | 16.3124 |

From the above table we see that as the learning rate is 0.3 there is the bigger Training and testing MAE which gives very bad results and as the learning rate is reduced the training and testing MAE has reduced and also started giving better results. Multilayer perceptron gives very low accuracy as it has many nodes in a layer and also as we know it is using back propogation that MLP is a type of feed-forward artificial neural network that generates a set of outputs from a set of inputs.

1. For MultiLayerPerceptron with one hidden Layer.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Hidden Nodes | Architecture | Parameters | Train MAE | Epochs | Test MAE | Overfitting |
| 1 | 5 | ii-hh-oo  (23-01-1) | Lr=0.3 | 16.4925 | 500 | 17.1613 | 0.67  NO |
| 2 | 5 | ii-hh-oo  (23-01-1) | Lr=0.5 | 21.4845 | 1000 | 19.3668 | 2.12  YES |
| 3 | 5 | ii-hh-oo  (23-01-1) | Lr=0.01 | 14.2259 | 2000 | 16.0156 | 1.79  YES |
| 4 | 5 | ii-hh-oo  (23-01-1) | Lr=0.02 | 13.2659 | 3000 | 15.8688 | 2.6  YES |
| 5 | 5 | ii-hh-oo  (23-01-1) | Lr=0.005 | 14.1733 | 4000 | 16.3443 | 2.17  YES |

We can see from the above results that when there are 5 hidden nodes it performed a lot better than the previous single layer perceptron. We also got to know that as the Learning rate is reduced the model is not performing well as we see that there is overfitting as there is difference between train and test MAE. Finally, MultiLayerPerceptron works better than Single-Layer Perceptron as it has many nodes in a layer and also as we know it is using back propogation that MLP is a type of feed-forward artificial neural network that generates a set of outputs from a set of inputs.

1. For Different Number of hidden nodes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Hidden Nodes | Architecture | Parameters | Train MAE | Epochs | Test MAE | OVERFITTING |
| 1 | 4 | ii-hh-oo  (23-01-1) | Lr=0.3 | 16.1747 | 500 | 17.0459 | NO |
| 2 | 6 | ii-hh-oo  (23-01-1) | Lr=0.3 | 15.4722 | 500 | 16.8883 | YES |
| 3 | 8 | ii-hh-oo  (23-01-1) | Lr=0.3 | 15.2365 | 500 | 17.7948 | YES |
| 4 | 10 | ii-hh-oo  (23-01-1) | Lr=0.3 | 15.139 | 500 | 17.1531 | YES |
| 5 | 13 | ii-hh-oo  (23-01-1) | Lr=0.3 | 13.4552 | 500 | 18.0257 | YES |

From the above results we can say that the best number of hidden nodes is 4.As the node goes on increasing there comes the overfitting. We can see that as the nodes increases the amount of overfitting also increases. When we increases the layers the train and test MAE will start giving the better results as the network becomes strong enough.

1. For one Layer with 5 Hidden Nodes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Hidden Nodes | Architecture | Parameters | Train MSE | Epochs | Test MSE | OVERFITTING |
| 1 | 5 | ii-hh-oo  (23-01-1) | Lr=0.3  Momentum=0.2 | 16.4925 | 500 | 17.1613 | 0.66  NO |
| 2 | 5 | ii-hh-oo  (23-01-1) | Lr=0.4  Momentum=0.3 | 19.5816 | 500 | 19.1117 | 0.046  NO |
| 3 | 5 | ii-hh-oo  (23-01-1) | Lr=0.6  Momentum=0.5 | 23.3681 | 500 | 21.8596 | 1.50  YES |
| 4 | 5 | ii-hh-oo  (23-01-1) | Lr=0.8  Momentum=0.7 | 83.0471 | 500 | 63.8991 | 19.148  YES |
| 5 | 5 | ii-hh-oo  (23-01-1) | Lr=0.01  Momentum=0.9 | 14.1444 | 500 | 16.0482 | 1.9038  YES |

When the learning rate 0.3 we see that the results were better but when the learning rate was 0.01 we got to know that it takes more time to load and also the results were not good. We also saw that as the learning rate was increased and the momentum was increased the training and testing error also started to increase which resulted in Overfitting. To conclude we can add more than one hidden layer so we can get the better improved results.

1. Table for 5 hidden nodes with no validation data.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Hidden Nodes | Architecture | Parameters | Train MSE | Epochs | Test MSE | OVERFITTING |
| 1 | 5 | ii-hh-oo  (23-01-1) | Lr=0.3  Momentum=0.2 | 16.4925 | 500 | 18.9992 | YES |

As we can see from the table that there are 5 hidden nodes and learning rate and momentum are default and there exist an overfitting. This happens because the model doesn’t know when to stop training as in properties we have stated validation threshold as 0.So when the training error has decreased and again after some time it increased the model starts resulting in overfitting. Unlike JavaNNS we can’t see error graph in WEKA we can only see the MSE and so we can’t stop the training data and thus it results in overfitting.

8. Absolute error is the sum of magnitude errors (actual - observed) of all instances. MAE considers the instances count to compare with absolute error which gives difference value. Whereas, RAE considers the actual values instead of instances count to compare with absolute error which gives the percent difference. Thus, RAE (percentage difference) is used for measuring prediction of any model as it is easier to interpret.

9. Table for Multilayer Perceptron

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run No | Hidden Nodes | Architecture | Parameters | Train MAE | Epochs | Test MAE | Overfitting |
| 1 | 5 | ii-hh-oo  (23-01-1) | Lr=0.3 | 16.4925 | 500 | 17.1613 | 0.67  NO |
| 2 | 7 | ii-hh-oo  (23-01-1) | Lr=0.5 | 15.9724 | 1000 | 16.6476 | 0.6752  NO |
| 3 | 9 | ii-hh-oo  (23-01-1) | Lr=0.01 | 13.837 | 2000 | 15.9311 | 2.0941  YES |

Table for M5P

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Run No | Classifier | Parameters | Train MSE | Test MSE | OVERFITTING |
| 1 | M5P | M =4  batchSize = 100 | 16.2756 | 15.4107 | 0.8649  NO |
| 2 | M5P | M = 6  batchSize = 100 | 16.345 | 15.6828 | 0.6622  NO |
| 3 | M5P | M =12  batchSize = 200 | 16.4793 | 16.0471 | 0.4322  NO |

As we can see from both the tables the MAE for train and test is less as compared to multilayer perceptron. We can see that almost all the results from M5P classifier are with no overfitting whereas the Multilayer perceptron has few parameters with overfitting.M5P is a binary regression tree model where the last nodes are the linear regression functions that can produce continuous numerical attributes. We can get the better results in Multilayer Perceptron by increasing the hidden layers as it has many nodes in a layer and also as we know it is using back propogation that MLP is a type of feed-forward artificial neural network that generates a set of outputs from a set of inputs.M5P works better on continuous numerical values as it is a binary regression tree model which does classification via regression. If there are more categorical values in a particular column then M5P doesn’t works and if class variable is categorical it won’t work.

**Part 3: Data Mining**

The movie data was collected from the IMDB web site which includes columns such as movie title, title year, duration, director name, actor name, user votes, critic reviews, gross, budget and imdb score, etc. The dataset when loaded had around 5000 instances with some missing values. We replaced all the missing values using weka filter function as the missing values were in small percentage with respect to the total instances. As there were around 20 columns we had to do feature engineering i.e. we used weka’s attribute selection (Best First) method to select the important attributes needed for our tasks.

We selected imdb score as a target variable as it was a best fit for measuring the greatness of any movie. As imdb score is a continuous numeric attribute we thought that the numeric prediction algorithms would work better in terms of performance. Classification algorithms wouldn’t work as most of the dataset includes numeric attributes. Similarly, clustering and association finding wasn’t used as our aim was about measuring the effectiveness of movies based on imdb score.

As we began using numeric algorithms like Linear regression for prediction the weka kept crashing. So we reduced to 8 no of attributes. When it comes to measuring the greatness of movie, apart from imdb score, a movie could get high gross through director popularity, actor popularity or country. There were more than 2500+ director names and several actor names which are why our weka kept on crashing when considering these attributes. We even used Discretization and NominaltoBinary filters on these attributes to encode the category values but still weka kept on crashing. Also, we used only one category attribute for e.g. Director Name or Movie Title and rest numerical attributes for building model. But, the Numerical algorithms took more than 10-15 minutes and yet model wasn’t build. The reason was too many category values to handle while training the data. Due to this reason we had to only use country as a nominal attribute (category) and discard the important features like director name, actor names, etc.

Following are the visualizations (golden nuggets/patterns/insights) of the data:-

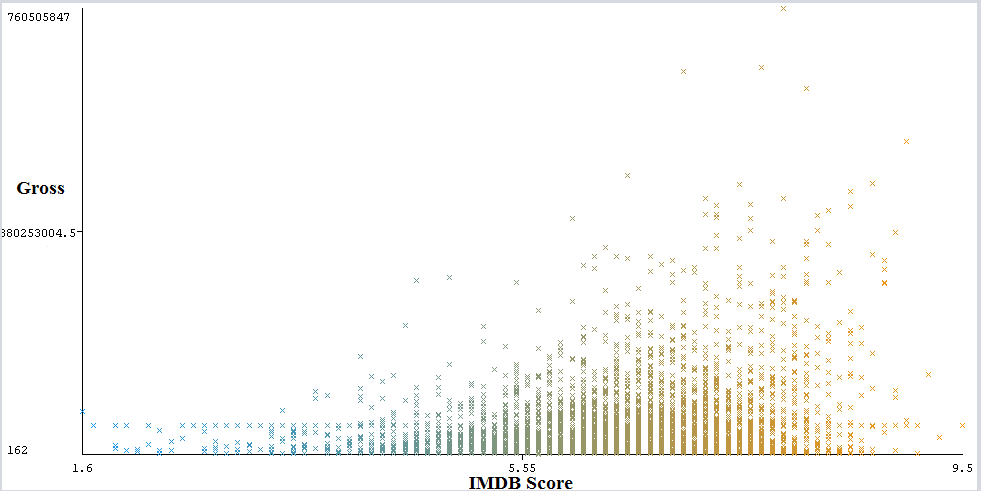


Fig 1

In the Fig 1, we can see that the lines of points are higher in between 6 to 8 IMDB Score. The gross value also increases for the 6 to 8 imdb scores. This means that as the gross value increases the imdb score is expected to be between 6 and 8.

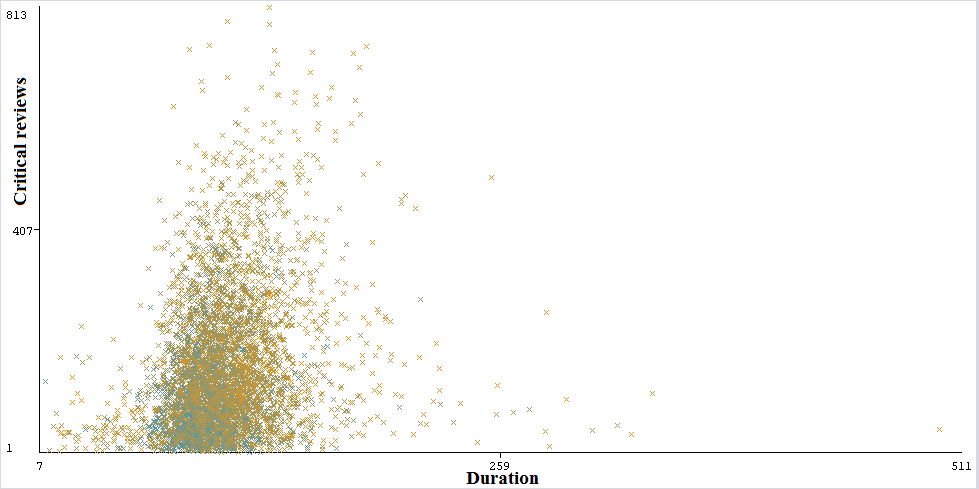


Fig 2

The blue points are IMDB scores between 1 and 6-5 whereas the brown points are IMDB scores between 6 and 9.9. In the Fig 2, we can see that for the duration between 60 and 124 (in minutes) the majority of the points are clustered together. The brown points (6-9.9 imdb score) tends to increase as the numbers of critical reviews increases. This means that if the critical reviews are higher the imdb scores are between 6 and 9.9 for the duration of the movies between 60-124 minutes.

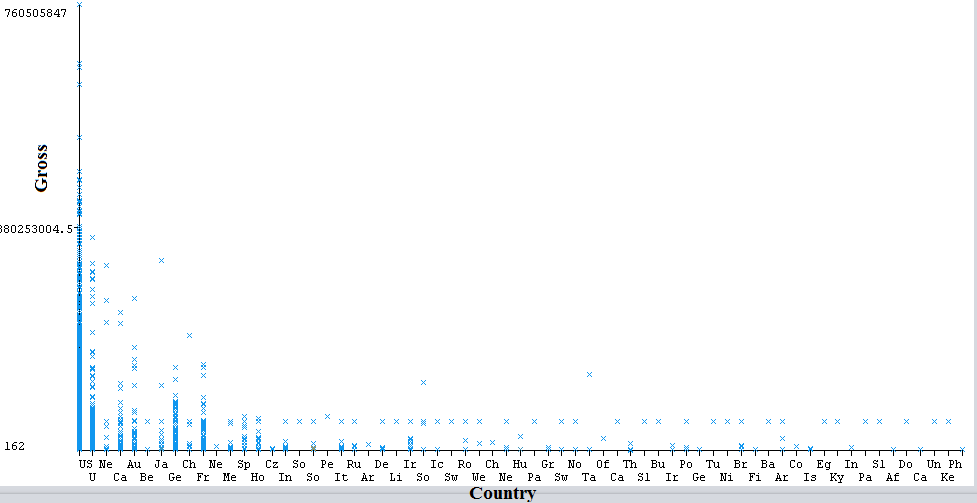


Fig 3

In the Fig 3, we can conclude that, US (United States), U (United Kingdom), Ca (Canada), Au (Australia), Ge (Germany) and Fr (France) are the top 6 countries with higher movie gross (in Dollars) values.

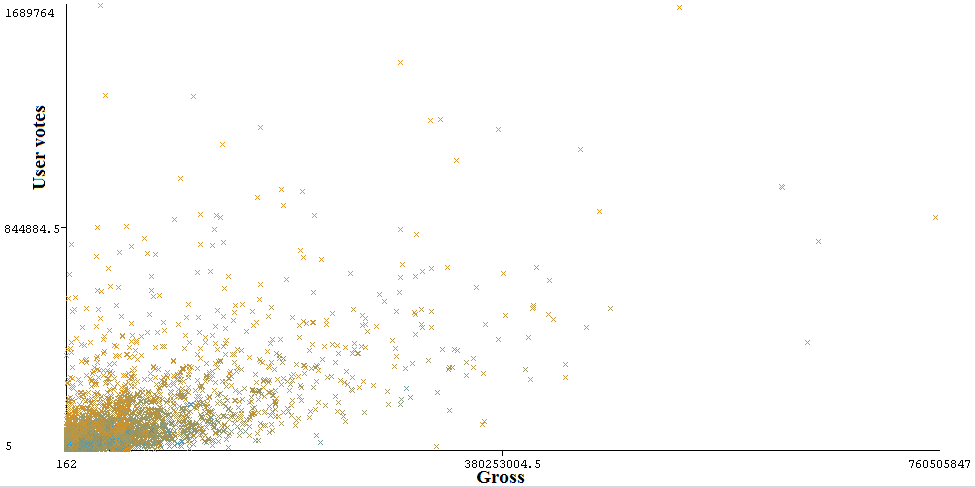


Fig 4

In the Fig 4, we can say that the line of points looks roughly linear. That means that as the gross value increases the User votes tends to increase too. Also, by combining all the visualizations we can conclude that Gross, User votes, Critical reviews and IMDB scores are closely correlated with each other. This also provides evidence that IMDB scores could be used as a target dependent variable since its more interpretable than the values of the other 3 features. Lets move on to predicting the model.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms** | **Training error (RAE)** | **Cross Validation error (RAE)** | **Overfitting** |
| IBk | 0% | 98.46% | YES |
| Random Forest | 25.67% | 69.35% | YES |
| M5P | 66.71% | 75.46% | YES |
| Linear Regression | 84.97% | 86.33% | NO |

We chose M5P and Linear Regression as both works on predicting Numerical class. In some cases, random forest could be used to classify the target variable. IBk was chosen as it works when more category values are to be used. In the above table, we can see that only Linear Regression algorithm is not overfitting while rests of the algorithms are overfitting. So we chose to perform hyper-parameter tuning on Linear Regression.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Parameters** | **Correlation Coefficient (For Training set )** | **Correlation Coefficient (For Cross Validation)** | **Training error (RAE)** | **Cross Validation Error (RAE)** | **Overfitting** |
| Linear Regression | batchSize = 50 | 0.49 | 0.47 | 84.97% | 86.33% | NO |
| Linear Regression | batchSize = 100 | 0.49 | 0.47 | 84.97% | 86.33% | NO |
| Linear Regression | numDecimalPlaces = 4 | 0.49 | 0.47 | 84.97% | 86.33% | NO |
| Linear Regression | batchSize = 400 | 0.49 | 0.47 | 84.97% | 86.33% | NO |
| Linear Regression | numDecimalPlaces = 20 | 0.49 | 0.47 | 84.97% | 86.33% | NO |

In the above table, we can say that despite changing the parameter values, the difference errors and coefficient values remains constant. We found that only the time taken for each run was different.

Finally, we can conclude that Linear Regression works better in terms of prediction compared to other models. This is because the target variable is a continuous numeric attribute where in general Linear Regression is used to predict the numerical values. As we mentioned earlier, we had to discard important features like Director Name, Actor Name, Movie title for the building an efficient model (avoid weka crash). Also, since the categorical values were too many for these attributes, visualizations in Weka using these attributes couldn’t be interpreted. The model could now only be able to predict score based on a country which includes lot of mislabelled errors.

**Pair work submission**

My Name is Mark Pereira and my other member name is Shonil Dabreo. We both worked as a pair. First of all I would like to say that we both contributed equally to this assignment. Firstly in classification with neural Networks we solved each question by running results into the Weka, while I was running the dataset on the Weka and running results my group member Shonil also noted all the calculations which were needed and later we both compared our results. Like these we try to complete the whole part 1. Now after completing part 1 to part 3 and discussing all the results we began with the report. For the part 4 we discussed about presentation. Some slides were made by me while some slides were made by shonil and finally both of us gave presentation by distributing the slides.

**References**

Data Mining Lecture Notes and Practicals and Tutorials Slides.