DATA MINING COSC 2111
Assignment 2

PART 1: CLASSIFICATION

3.1.1: Thypothyroid.arff is loaded and observered that TBG has no data values. Therefore, TBG column can be removed as there is nothing to do with the variable. The missing values of the nominal are replaced with the modes of that attribute and the numeric attributes by their corresponding means. This can be achieved by applying "replacemissingvalues" in filter. ("Filter -> Unsupervised -> Attribute -> replacemissingvalues"). Now the data is Normalized as there is high variance in the observations of attributes. If the data is not normalized the attributes with higher value observations will influence the prediction results of the classification or the classifier making if to give false or misleading reports. ("Filter -> Unsupervised -> Attribute -> Normalized"). We can also convert nominal and numeric attributes to binary if necessary. Here, let us convert nominal to binary. But this can be done when we build the neural network by setting the "nominalToBinaryFilter" to "True" in the "MultilayerPerceptron". For nominal attributes say for example sex, which has 2 attributes male and female. Male is encoded as "0 1" and female as "1 0" when 2 input nodes are considered or 0 for female and "1" for male if "1" input node is considered. Likewise, each nominal attribute will have inputs as many as the factors or labels of that nominal attribute. Where-as, numerical attributes will have only 1 input. Therefore, neural network on target class variable will have, "Number of Inputs – 33 (Num of Input nodes), Number of outputs – 4 (Num of Output nodes)".

3.1.2: Pre-Processing procedure to generate training, validation, and test data files in Weka:
Let us split "hypothyroid.arff" dataset into 60% training set, 20% validation set and 20% test set. Load the "hypothyroid.arff" dataset into Weka and select Preprocess tab and go to Filter -> Unsupervised -> Instances -> Resample and then set noReplacement to True, such that the training set, validation set, and test set does not contain any duplicate values. Set sampleSizePercent to 60 and click ok. Next, click on Apply button to generate 60% (2263 instances) train set from overall 3772 instances. Click on save button to save training set as .arff file. To generate validation and test sets, click on Undo button to go back to original 3772 instances. Now set "invertSelection" to True to obtain remaining 40% data. Select Apply button, such that remaining 1509 (40%) instances will generate. Now split obtained 40% instances into 50% validation and 50% test data. Select Resample and set invertSelection to False and set sampleSizePercent to 50. Now click on apply button to generate 754 validation data instances. Click on save button to save validation set as .arff file. Now click on Undo button to go back to original 40% instances and then select InvertSelection to True and click on ok and then select apply button to obtain 755 test instances. Click on Save button to save test data set as .arff file.

3.1.3: If unknown class label comes into a trained neural network, the neural network predicts it as one of the class labels by checking the closest class label. Which will be misclassified, and confidence also comes down, which makes it difficult to filter out.

3.1.4: Increasing the number of epochs on a single layer perceptron or 0 – hidden layer networks increase both train and test errors with a simultaneous increase in overfitting. Even though it performs better at the initial level, it fails to perform when the complexity increases. Here, when the epochs increased the complexity increased and there by the network learned more on the same data. More precisely data is learnt in a single procedure repeated for all epochs. Therefore, more the complexity is, the more the layers (hidden layers) are required. Therefore, multilayer perceptron is more efficient when the problem is complex.

Run No	Architecture	Parameters	Train Root MSE	Train MSE	Train Error (%)	Epochs	Test Root MSE	Test MSE	Test Error (%)	Over fitting
1	**33 - 0 – 4**	**LR = 0.3**	**0.1556**	**0.0242**	**5.1259**	**500**	**0.1666**	**0.0278**	**5.9603**	**0.8344**
2	**33 - 0 – 4**	**LR = 0.3**	**0.1556**	**0.0242**	**5.1259**	**1000**	**0.1674**	**0.028**	**5.9603**	**0.8344**
3	**33 - 0 – 4**	**LR = 0.3**	**0.1562**	**0.0244**	**5.0817**	**1500**	**0.1679**	**0.0282**	**5.9603**	**0.8786**
4	**33 - 0 – 4**	**LR = 0.3**	**0.1567**	**0.0246**	**5.0817**	**2000**	**0.1681**	**0.0282**	**5.9603**	**0.8786**
5	**33 - 0 – 4**	**LR = 0.3**	**0.1569**	**0.0246**	**5.0817**	**2500**	**0.1681**	**0.0282**	**5.9603**	**0.8786**

3.1.5: For a 4 – node – single hidden layer network the optimal values are observed when epochs are 2000 (i.e., in the 4th run). Even though 2500 shows the lower over fitting value but the train and test errors are more. Also, we can clearly observe that, with the addition of hidden layer, the overfitting decreased when the epochs are increased as discussed in 3.1.4.

Run No	Architecture	Parameters	Train Root MSE	Train MSE	Train Error (%)	Epochs	Test Root MSE	Test MSE	Test Error (%)	Over fitting
1	**33 - 5 – 4**	**LR = 0.3**	**0.1135**	**0.0129**	**3.8445**	**500**	**0.1634**	**0.0266**	**6.755**	**2.9105**
2	**33 - 5 – 4**	**LR = 0.3**	**0.1254**	**0.0157**	**3.7561**	**1000**	**0.1555**	**0.0241**	**5.8278**	**2.0717**
3	**33 - 5 – 4**	**LR = 0.3**	**0.1082**	**0.0117**	**2.7839**	**1500**	**0.1325**	**0.0176**	**4.2384**	**1.4545**
4	**33 - 5 – 4**	**LR = 0.3**	**0.0717**	**0.0051**	**1.3257**	**2000**	**0.1047**	**0.0109**	**2.649**	**1.3233**
5	**33 - 5 – 4**	**LR = 0.3**	**0.0917**	**0.0084**	**2.2095**	**2500**	**0.1193**	**0.0142**	**3.0464**	**0.8369**

3.1.6: It is observed that for 10 hidden nodes we got the least test error as well as the test MSE along with minimal overfitting value. Even though Train MSE and train error are least for 5 – node and 20 - node, their overfittings are more. Therefore, 10 – hidden layers might be better in this case. In case of 25 hidden nodes the network learnt lesser than the required (since negative sign). Hence suggesting underfitting.

Run No	Architecture	Parameters	Train Root MSE	Train MSE	Train Error (%)	Epochs	Test Root MSE	Test MSE	Test Error (%)	Over fitting
1	**33 - 5 – 4**	**LR = 0.3**	**0.1235**	**0.0152**	**3.8445**	**500**	**0.1634**	**0.0267**	**6.755**	**2.9105**
2	**33 - 10 – 4**	**LR = 0.3**	**0.1238**	**0.0153**	**3.7561**	**500**	**0.1552**	**0.0241**	**5.298**	**1.5419**
3	**33 - 15 – 4**	**LR = 0.3**	**0.1273**	**0.0162**	**3.8445**	**500**	**0.165**	**0.0272**	**6.6225**	**2.778**
4	**33 - 20 – 4**	**LR = 0.3**	**0.1242**	**0.0154**	**3.5351**	**500**	**0.1671**	**0.0279**	**6.4901**	**2.955**
5	**33 - 25 – 4**	**LR = 0.3**	**0.1208**	**0.0146**	**3.6235**	**500**	**0.1188**	**0.0141**	**3.5242**	**-0.0993**

3.1.7: Train and Test runs are executed on 60% train data and 20% test data by varying learning rate and momentum with five hidden nodes and 500 epochs. Model performs well with lr=0.4 and m=0.1 by less overfitting of 1.1882 compared with other combinations. Therefore, high learning rate and low momentum with significant difference gives the best results.

3.1.8: J48 has more accuracy compared to Multilayer Perceptron. It got 100% accuracy on test data and 99.8 % approx. 100% on train data.

Classifier	Train Accuracy (%)	Test Accuracy (%)	Pros	Cons
J48	**96.1555**	**93.5099**	Easy to interpret Visualisation with a Decision tree	Complex on large data
Multilayer Perceptron	**99.7791**	**100**	Element failure doesn't affect execution due to parallel nature. It learns hence it doesn't require any reprograming.	Emulation is required due to its architecture. High processing time on a large network.

3.1.9: Multilayer Perceptron is 10 to 2000 times slower than other methods and JavaNNS can be preferred. But for me Weka is more handy than JavaNNS, as in JavaNNS setting up environment is more complex and the process is some what different to me. Therefore, I used Weka Multilayer Perceptron.

PART 2: NUMERIC PREDICTION

3.2.1: The heart-v1.arff dataset is initially loaded into weka and checked for missing values. The missing values are now replaced with the mean and mode values using "replacemissingvalues" filter. As the target variable is "chol". We need to make it the predictive class. This can be done by selecting the edit button and then right click on the chol variable and select "Attribute as Class". The data can be normalised using normalize filter. The nominal attributes can be converted to binary using "nominalToBinaryFilter" filter if necessary. For nominal attributes say for example sex, which has 2 attributes male and female. Male is encoded as "0 1" and female as "1 0" when 2 input nodes are considered or 0 for female and "1" for male if "1" input node is considered. Likewise, each nominal attribute will have inputs as many as the factors or labels of that nominal attribute. Where-as, numerical attributes will have only 1 input. Therefore, neural network on target class variable will have, "Number of Inputs – 25 (Num of Input nodes), Number of outputs – 1 (Num of Output nodes)".

3.2.2: Load "heart-v1.arff" into Weka and pre-process the data set to generate 60% training set, 20% validation set and 20% test set. By applying the steps listed in PART 1, we have generated 363 instances as training set, 121 instances as validation set and 122 instances as test set out of 606 instances.

Mean Absolute Error (MAE) is the average of all absolute errors, and it is calculated using the formula,

$$MAE = \frac{\sum\sum_{i=1}^{n}|x_i - \bar{x}|}{n}$$

The Numeric input attributes must be scaled or normalized to reduce the influence of higher value observations to get true predictions. But reverse scaling the neural network outputs helps in predicting the accurate results.

3.2.3: Train and Test runs are executed on 60% train data and 20% test data by varying number of Epochs with no hidden layer. On analysing MAE, training errors are decreased gradually, and test errors are increased. Therefore, overfitting increased significantly on increasing Epochs. For solving simple problems single layer perception performs well and for the complex problems multilayer perception can be used but it takes more computation time compared to single layer perception.

Run No	Architecture	Parameters	Train Root MSE	Train MSE	Train MAE	Epochs	Test Root MSE	Test MSE	Test MAE	Over fitting
1	**25-0-1**	**LR = 0.3**	**0.0447**	**0.002**	**0.0362**	**500**	**0.0729**	**0.0053**	**0.0554**	**0.0192**
2	**25-0-1**	**LR = 0.3**	**0.0372**	**0.0014**	**0.0292**	**1000**	**0.0863**	**0.007**	**0.0576**	**0.0284**
3	**25-0-1**	**LR = 0.3**	**0.0349**	**0.0012**	**0.0272**	**1500**	**0.1095**	**0.012**	**0.064**	**0.0368**
4	**25-0-1**	**LR = 0.3**	**0.0309**	**0.0009**	**0.0239**	**2000**	**0.1151**	**0.0132**	**0.0669**	**0.043**
5	**25-0-1**	**LR = 0.3**	**0.0307**	**0.0009**	**0.0236**	**2500**	**0.1181**	**0.0139**	**0.0706**	**0.047**

3.2.4: Train and Test runs are executed on 60% train data and 20% test data by varying number of Epochs with one hidden layer and five hidden nodes. Train and Test errors are increased significantly till four runs with a variation of 500 epochs. As, a result overfitting increases on increasing epochs. On 2500 epochs train and test errors decreased slightly but there is no change in the increase of overfitting. Overfitting is less with hidden layer as compared with no hidden layer. As a result, Model is learning more with one hidden layer than no layer.

Run No	Architecture	Parameters	Train Root MSE	Train MSE	Train MAE	Epochs	Test Root MSE	Test MSE	Test MAE	Over fitting
1	**25 - 5 – 1**	**LR = 0.3**	**0.0797**	**0.0064**	**0.0684**	**500**	**0.0887**	**0.0077**	**0.0736**	**0.0052**
2	**25 - 5 – 1**	**LR = 0.3**	**0.0821**	**0.0067**	**0.0708**	**1000**	**0.0925**	**0.0086**	**0.0764**	**0.0056**
3	**25 - 5 – 1**	**LR = 0.3**	**0.0835**	**0.007**	**0.0726**	**1500**	**0.0964**	**0.0093**	**0.0802**	**0.0076**
4	**25 - 5 – 1**	**LR = 0.3**	**0.0843**	**0.0071**	**0.0738**	**2000**	**0.1001**	**0.01**	**0.083**	**0.0092**
5	**25 - 5 – 1**	**LR = 0.3**	**0.0821**	**0.0067**	**0.0718**	**2500**	**0.1005**	**0.0101**	**0.0817**	**0.0099**

3.2.5:

On checking with different number of hidden nodes, train and test errors are low with less overfitting of 0.0006 on one hidden node. Hence, 1 hidden node is good for this problem.

Run No	Architecture	Parameters	Train Root MSE	Train MSE	Train MAE	Epochs	Test Root MSE	Test MSE	Test MAE	Over fitting
1	**25 - 1 – 1**	**LR = 0.3**	**0.0522**	**0.0027**	**0.0402**	**500**	**0.0506**	**0.0026**	**0.0408**	**0.0006**
2	**25 - 5 – 1**	**LR = 0.3**	**0.0797**	**0.0064**	**0.0684**	**500**	**0.0887**	**0.0079**	**0.0736**	**0.0052**
3	**25 - 10 – 1**	**LR = 0.3**	**0.047**	**0.0022**	**0.0384**	**500**	**0.0786**	**0.0062**	**0.0614**	**0.023**
4	**25 - 15 – 1**	**LR = 0.3**	**0.0361**	**0.0013**	**0.0282**	**500**	**0.0746**	**0.0056**	**0.056**	**0.0278**
5	**25 - 20 – 1**	**LR = 0.3**	**0.042**	**0.0018**	**0.0335**	**500**	**0.0758**	**0.0057**	**0.0581**	**0.0246**

3.2.6: Train and Test runs are executed on 60% train data and 20% test data by varying learning rate and momentum with five hidden nodes and 500 epochs. Model performs well with high learning rate and less momentum with a difference of 0.1. Values with lr=0.4 and m=0.3, predicted less overfitting of 0.0031.

Run No	Architecture	Parameters	Train Root MSE	Train MSE	Train MAE	Epochs	Test Root MSE	Test MSE	Test MAE	Over fitting
1	**25-5-1**	**LR = 0.2, M = 0.2**	**0.0519**	**0.0027**	**0.0425**	**500**	**0.0673**	**0.0045**	**0.0553**	**0.0128**
2	**25-5-1**	**LR = 0.3, M = 0.2**	**0.0797**	**0.0064**	**0.0684**	**500**	**0.0887**	**0.0079**	**0.0736**	**0.0052**
3	**25-5-1**	**LR = 0.2, M = 0.3**	**0.0537**	**0.0029**	**0.0439**	**500**	**0.0708**	**0.005**	**0.0576**	**0.0137**
4	**25-5-1**	**LR = 0.4, M = 0.1**	**0.06**	**0.0036**	**0.049**	**500**	**0.0745**	**0.0056**	**0.0587**	**0.0097**
5	**25-5-1**	**LR = 0.4, M = 0.3**	**0.0849**	**0.0072**	**0.0727**	**500**	**0.0895**	**0.008**	**0.0758**	**0.0031**

3.2.7: Runs executed with five hidden nodes and identified from 5000 Epochs there is no change in the values of train and test errors. For 5000, 5100 and 5200 epochs train and test errors are recorded as 0.0502 and 0.061 respectively, as a result overfitting is recorded as 0.0108. With minimum number of epochs, we can identify less overfitting than a greater number of epochs.

3.2.8: Both Relative Absolute Error (RAE) and Mean Absolute Error (MAE) are used to measure the performance of a predictive model. Relative Absolute Error is the ratio, comparing a mean error to errors produced by naïve model. Whereas Relative Absolute error gives the average of all absolute errors. Relative Absolute Error is calculated on two different units whereas Mean Absolute Error is calculated on single unit (i.e., say for example - on the values of same attribute.). So, we prefer Mean Absolute Error to predict the performance of a model.

Classifier	Train MAE	Test MAE	Pros	Cons
M5P	**0.0597**	**0.0554**	Induces trees of regression models Visualisation with a Decision tree	Not easily solvable.
Multilayer Perceptron	**0.0369**	**0.0354**	Element failure doesn't affect execution due to parallel nature. It learns hence it doesn't require any reprograming.	Emulation is required due to its architecture. High processing time on a large network.

3.2.9:

M5P has more absolute error compared to Multilayer Perceptron. This makes Multilayer Perception to choose over M5P.

PART 3: DATA MINING

3.1: INTRODUCTION

"The movie data was collected from the IMDb web site which claims to be "the world's most popular and authoritative source for movie, TV and celebrity content".
5043 observations, 37 Attributes – 27 Nominal and 10 Numeric.

3.2: BODY

3.2.1: QUESTION

How can we tell the greatness of a movie before its release?

3.2.2: DATA PREPARATION, PREPROCESSING

The data is initially loaded into weka and checked for missing values. The missing values are now replaced with the mean and mode values using replacemissingvalues filter. If required, the data can be normalised using normalize filter. The nominal attributes can be converted to binary using nominaltobinary filter if necessary. The data is prepared, and further pre-processing is done for association finding. The numerical attributes are removed as apriori algorithm does not work on numerical attributes.

3.2.3: DATA MINING TECHNIQUE

3.2.3.1: CLASSIFICATION – DECISION STUMP. Load data set to Weka and then select classifier -> DecisionStump -> choose training set -> select attribute -> start run and analyse the results.
3.2.3.2: CLUSTERING – KMEANS. Now select clustering -> simpleKmeans -> choose training set -> select attribute -> start run and analyse the generated results.
3.2.3.3: ATTRIBUTE SELECTION.
Attribute Evaluator – CfsSubsetEval (Parameters – "P = 1, E = 1")
Search Method – BestFirst (Parameters – "D = 1, N = 5")
By running cross validation with 10 folds by selecting the above methods we got all the attributes.
3.2.3.4: ASSOCIATION FINDING – APRIORI ALGORITHM. The numeric attributes are removed as Association does not support on the numeric data. The nomial attributes are sent to association finding and the rules are obtained with 85% Support and approx. 98% confidence.
3.2.3.1: VISUALIZATIONS. Select the visualization tab and analyse the required attributes by generating different pictorial representations.

3.2.4: RESULTS

3.2.3.1: CLASSIFICATION - On analysing different classifier models, Decisionstump shows the best results among all with no Overfitting and less training and testing errors. Therefore, for this "IMDB-movie-data.csv" dataset Decisionstump classifiers predicts more accurately with 3.9064 training and testing errors than other classifiers. Although, ZeroR shows no overfitting but the error rate is high compared to DecisionStump.

Classifier	Training Root MSE	Training MSE	Train Error (%)	Test Root MSE	Test MSE	Test Error (%)	Over-Fitting
DecisionStump	0.1842	0.033	3.9064	0.1844	0.034	3.9064	0
ZeroR	0.1993	0.039	4.1444	0.1993	0.039	4.1444	0
IBK	0.0002	0	0	0.2422	0.058	5.8695	5.8695
J48	0.1842	0.033	3.9064	0.1845	0.034	4.0452	0.1388
OneR	0	0	0	0.2016	0.04	4.065	4.065
NavieBayes	0.3977	0.158	20.5037	0.5788	0.335	41.5229	21.0192

3.2.3.2: CLUSTERING - k-Means is faster and divided the instances into two clusters with 49% and 51% respectively within 0.03 seconds. Even though EM is slow it can handle both Nominal and Numeric data. So, it depends on the type of data chosen and the person or the decision maker himself. Here,

for these clusters I prefer K-Means as it is faster and works based on Euclidean distance, whereas EM works on density probability. Also, in EM based on membership probability, an instance belongs to many clusters, whereas in K-Means an instance is the member of the single cluster.

3.2.3.4: ASSOCIATION FINDING

Support: 85%

Confidence: 98%

```
Best rules found:

 1. color=Color language=English 4529 ==> Documentary=f 4427    <conf:(0.98)> lift:(1) lev:(0) [6] conv:(1.06)
 2. language=English 4716 ==> Documentary=f 4607    <conf:(0.98)> lift:(1) lev:(0) [4] conv:(1.03)
 3. color=Color 4834 ==> Documentary=f 4720    <conf:(0.98)> lift:(1) lev:(0) [1] conv:(1.01)
 4. Mystery=f 4543 ==> Documentary=f 4423    <conf:(0.97)> lift:(1) lev:(-0) [-10] conv:(0.9)
 5. Family=f 4497 ==> Documentary=f 4378    <conf:(0.97)> lift:(1) lev:(-0) [-11] conv:(0.9)
 6. Horror=f 4478 ==> Documentary=f 4357    <conf:(0.97)> lift:(1) lev:(-0) [-13] conv:(0.88)
 7. Sci-Fi=f 4427 ==> Documentary=f 4307    <conf:(0.97)> lift:(1) lev:(-0) [-13] conv:(0.88)
 8. Documentary=f language=English 4607 ==> color=Color 4427    <conf:(0.96)> lift:(1) lev:(0) [10] conv:(1.05)
 9. language=English 4716 ==> color=Color 4529    <conf:(0.96)> lift:(1) lev:(0) [8] conv:(1.04)
10. Documentary=f 4922 ==> color=Color 4720    <conf:(0.96)> lift:(1) lev:(0) [1] conv:(1)
```

For example, Rule 1

(“Color movies are mostly English movies but not documentary”) can be predicted with 98% confidence.

3.2.4.4: VISUALISATION

Most of the movies are action, comedy, Drama, Romance, Thriller. Documentary oriented movies are very less which are 121 movies only. As per IMDB-movie-data Most of the high budgeted English movies had been taken in USA, where English can be understood by most of the people in the world but first 2 high budget movies are from south korea. When we compare the cast_total_facebook_likes the USA has more likes than others. Not only cast_total_facebook_likes but also movie_facebook_likes are also more for USA. Now-a-days, all the movies are in color but in 2015 and 2014 there are 2 black and white movies. The 2 black and white movies which was taken in 2014 & 2015 are the last after that there are no black and white movies. Coming to IMDB_score the Canada has more like 9.5 but over-all the USA has good rating. The directors who had taken Hollywood movies (USA) has more facebook rating and most of the directors are from USA. The highest gross collection is also for USA movies.

3.2.5: CONCLUSIONS

I would like to conclude that most of the films made by US directors are coloured and in English, and I have found that we cannot forecast the film depending on the budget, stars, directors, since it failed in certain instances. However, it works in most situations and we should not judge the film by ranking or Facebook likes because it was explicitly reported that the movies with low ratings and likes have collected enormous collections. We can understand the progress rate by seeing the votes given by users. Suppose say, the "Avatar" movie directed by James Cameron in the USA in 2009 with $2370,000,000 was a mystery, Adventure and Science Fiction movie with just 33,000 Facebook movies, but it was a massive hit movie that raised $760,505,847 worldwide. The director has 0 likes and aspect ratio here was 1.7 and the Facebook likes were also less for actors. Also, I have observed that for the USA movies had collect the huge profit because when we see the movie “The Exorcist” the budget they spend around 8 million but the gross market was 204 Million.

3.2.5.1: Golden Nuggets

- Lesser the number of likes and lower is the rating, the more success is the movie.
- The directors who had taken Hollywood movies (USA) has more facebook rating and most of the directors are from USA.
- The highest gross collection is also for USA movies.

3.2.6: OVERALL CONCLUSIONS

Decision Stump classifier and K – means Cluster helped in predicting the most popular class and cluster of the dataset. Where-as visualisation and Apriori algorithm gave the findings for the golden - nuggets. From the conclusions and golden - nuggets we can clearly state the greatness of the movies before their release. The movies that are Hollywood movies and are directed by the directors of USA are more likely to get higher collections and become great movies.

Team Members:	**Contribution**
1. Sri Venkata Manideepu Maddipati (S3820822)	50%
2. Vishal Patnaik Damodarpatruni (S3811521)	50%

We both worked equally on each part of the assignment. We shared our knowledge on each concept of Data Mining covered in the assignment and did our best in answering each question.