CSE 572 DATA MINING

PROJECT REPORT

TEAM MEMBERS

Jose Ibarra 1202994242

Ajay Modi 1208526731

INTRODUCTION

Data Mining is the process of analyzing data from different sources and finding relationships not previously discovered or summarizing the data into useful information.[1] For example, the information can be used to increase revenue and cuts costs for a company. Technically, data mining is the process of finding correlations or patterns among the features in large data sets. On a higher level, this helps to determine relationships between internal and external factors and enables a company to make wise decisions, forecast the growth on the sales, and increase customer satisfaction. Data mining tasks can be divided into several steps: selection, data preprocessing, transformation, model selection, knowledge discovery.[2] In this project, we were given a large dataset with large number of features. Our objective was to classify the data correctly with the above mentioned data mining process.

The classification process consists of a model or a classifier which predicts the categorical labels. For example, predicting if the weather on a particular day will be sunny, rainy or cloudy. [3]

Many libraries, coding languages, and software exist for different machine learning and data mining tasks, of which we used the Java programming language to write scripts and WEKA software for the classification of records.[4] We used scripts written in Java for dimensionality reduction and to preprocess data. We used WEKA, which is a collection of machine learning algorithms, for data mining tasks to find the best classifier algorithms based on cross validation techniques on training data. WEKA also contains tools for data pre-processing, classification, regression, clustering, association rules, and data visualization.

LITERATURE REVIEW

The main aim of the project was to select features from the given data set, preprocess the training data, transform it into a WEKA accepted format, select the classification model, optimize the parameters for the selected model, train the model, and apply the same tuned model on the test dataset to predict the class label. The goal was to achieve best the performance in terms:[2]

- Accuracy – Accuracy of the classifier refers to the potential or ability of giving the correct output. Given test data, it gives us the measurement to compare a model with another model and how well the model fits on a particular dataset. In the case of classification, it can be referred as the number of correctly classified records divided by the total number of records.
- Speed – This refers to the computational cost involved in performing various data mining tasks mentioned.
- Robustness – Refers to how well a classifier handles noisy data or missing values.
- Scalability – Refers to the ability to construct the classifier efficiently, so it can work with large data sets.
- Interpretability- Refers to what extent the classifier understands inherent properties or relation of features.

OUR APPROACH

The goal of our classification was to accurately predict the target class for each case in the data. The classification task began with a data set in which the class assignments are known (training data) and based on training data. Our model learned the parameters and tuned itself with those parameters. Next time, when test data was given to this learnt model, these parameters carry out the functionality of predicting the correct class for the given test records.

As a first step to approaching the given problem, we started with understanding the data and whether we could see any relation in between the features and labels.[3] Are these features independent and identically distributed, or are they correlated? What types of features are given? (Numeric, string, nominal…) Do all the features have similar data values? (Uniformly distributed data, sparse data, heavily weighted values in data, unary data) Is the data missing values? Does it have noise, outliers, or duplicate data? The above task was performed by writing a script in Java to check whether there were sudden abrupt changes in values that might due to noise. Also, there are no missing values, certain data points which reside at the very edge of the distribution can be considered as outliers and we did not find any duplicate data. The data contained many attributes with only unary features, especially within the testing set.

After that, we had to select a proper software tool or programming language, which could help us in model selection. There were a number of available options out there like Orange, WEKA, GNU Octave, MatLab libraries, SciPy, etc. [5][4][6][7][8]We found WEKA, which was a sufficient tool to help us solve the given problem. Hence, we started understanding different functionalities and features that WEKA offered.

Once, we understood the different functionalities of WEKA, we started with applying proper filters to preprocess the data. Once data was preprocessed, since our dataset has 3400 features, we started exploring different feature selection methods and dimensionality reduction processes which could help us to remove irrelevant features or negligibly related features. In a nut shell, we wanted to reduce the size of the dataset. After evaluating each attribute evaluator and search method, we found that the filter attribute evaluator with ranker search method gave us the ranking of features in descending order based on correlation of features or how importantly they affected the outcome of classifier which helped us to reduce the number of features.

With the given ranking of the features by attribute selection method, we carried out the data mining task to check the accuracy of the random forest model with default parameters. We used the cross-validation technique to evaluate the performance. The results were:

Number of Attributes from Ranking Method	**Accuracy**
20	81.8
44	82.53
68	84.46
88	85.4
99	86.48
140	87

Table 1 - Accuracy of Ranking Method

As per Table 1, we found local maxima of accuracy at 87% with 140 features.

We researched the effectiveness of a principle component analysis (PCA) on our data set. A PCA on a data set attempts to reduce the dimensionality of the data by transforming the data set into a smaller set of uncorrelated principle components that represents the variation in the entire data set (PCA citation).[9] We conducted our PCA utilizing the Matlab dimensionality reduction toolbox that includes many dimensionality reduction methods, including PCA.[7] After conducting a PCA of the data, we inputted the PCA data set into a WEKA random forest model with default parameters. We used cross-validation to evaluate the performance of the model.

Number of Attributes from PCA	**Accuracy**
1500	67.5%
1000	72.87%
500	75%
250	74.3%
100	73%

Table 2 - Accuracy of PCA

Table 2 represents the accuracy obtained based on the number of attributes in the resulting PCA data set. PCA appeared to have a maximum of 75% using our PCA method, which was much lower than the results of the ranking method.

We analyzed the data manually to detect if there were any attributes that could be removed through a simple script. The training and test data contained attributes of only unary data. In total, the data sets contained 484 attributes that represented unary data. We removed these attributes from the training data and tested it with the random forest method. The default random forest ran on the training data set with the unary attributes removed had an accuracy of 87%. The manual analysis of the data was simpler than the ranker and PCA methods in terms of implementation. The manual analysis produced results as accurate as the ranker method and better than the PCA method.

Now we have reduced the size of data, we needed to convert class labels of training data to a nominal attribute to make it compatible for model selection in WEKA. Once the data was transformed to a compatible format, we selected different models to identify which model tunes the best with the given training data. By using the cross-validation technique, we were able to check the accuracy or performance of the model on the training data. Cross-validation ensured we made sure to sample data properly, so we could have a good representation of overall data in the training and test set.

Classification Model	**Accuracy on 5 Fold Cross-Validation**
Random Forest	86%
Bayesian Logistic	77%
J48	80%
Adaboost with Decision Stump	76%
Bagging	79%
Voted Perceptron	51%
IK3	56%

Table 3 - Accuracy of Classification Models

Based Table 3we decided to go with the random forest model. Random forest is an ensemble classification method.[10] Random forests create many classification trees based on the data. The individual classification trees are built as follows:

1) The tree is given a random sample of the data with replacement to train the tree.
2) Each node of the tree works with a specified number of attributes randomly selected from the set of attributes. The number of attributes a node can learn from is less than the total number of attributes in the data set. The node will split based on the selected set of attributes.
3) Each tree is maximally grown, with no pruning.

When classifying a record, each classification tree votes for a class based on its individual classification results.[10] The class with the most votes within the forest becomes the classification of that record.

We first trained the randomforest model with the default WEKA parameters which are max depth of 0, number of features of 0, number of trees of 100, and a seed of 1; the data set inputted was not preprocessed. Our accuracy ranged around 86% on cross validation. We tried to fine tune the parameters to improve accuracy further. We found that the square root of total number of features provided a good base point to optimize the parameters of random forest.[11] The random forest classification method worked best with the number of features set to 61 when inputting the unprocessed training data set. Also, we found that the seed of 3 and number of trees of 500 gave the best outcome in terms of accuracy and the lowest out of bag error.

ENCOUNTERED PROBLEMS AND SOLUTIONS

1) Training data was given into two separate files. One contained training records, the other contained training labels. Hence, many WEKA model selection methods and preprocessing techniques won't work until we merge both the data into a CSV file and then give that file as an input to WEKA.

2) We were not familiar with the WEKA software tool. At the beginning of the project, we had a few problems understanding all of the functions of the tool, the procedures, and the commands to execute the classifier algorithms. Online tutorials, online postings, forums and documentation on WEKA tools helped to understand more about WEKA. Continuous studying also expanded our knowledge about the tool more.

3) Many select attribute methods restricted on working on a particular type of data set and with particular search algorithms. For example, filtered attribute evaluators only worked with ranker search method and filtered subset evaluator only worked with greedy stepwise method.

4) While data preprocessing, class labels of the training records needed to change from nominal attributes from numeric attributes. This is because many of the classification modeling techniques only worked if the class labels are nominal.

5) After a certain point, increasing the training accuracy could be detrimental to the classification. If the training errors reduced but test error increased, that resulted in model overfitting. It is very difficult to find a point when to stop improving accuracy of training model. We tried to remain careful and never used test set data to train our model, so that our model never overfits.

6) Since the features didn't reveal any information at a first glance because of their nature (like feature 1, feature 2…, feature 3400), we couldn't use any domain knowledge to reduce the dimensionality of the feature. Rather, we had to go with different feature selection methods and different searching algorithms to figure out the best combination of features for our model.

7) When testing our model with votedperception, we faced an out of heap memory issue which says that WEKA used our max heap size to store the result during the calculation. If some model takes too much computational storage, eventually WEKA runs out of heap memory. To resolve it, we have to allocate extra memory from RAM to WEKA.

CONCLUSION

We achieved good accuracy for our model; in our research we found most classification methods have a maximum of around 90% accuracy.[12][13][14] Our model is robust since we resolved data quality issues during data preprocessing. It also takes less time to compare to other models to learn the parameters that best fit the given data. Also, since this approach can be carried out with large datasets, it is scalable and, with the help of histograms, we can also see the interpretability of the classifier.

For our submissions, we used the preprocessed data sets, classification method and parameters for the classification method shown in Table 4.

Preprocessed Data Set	**Classification Method**	**Parameters**
Original Data Set	Random Forest	numFeatures: 61 numTrees: 500 Seed: 3
Unary Data Removed	Random Forest	numFeatures: 56 numTrees: 500 Seed: 1

Table 4 - Project Submissions

REFERENCES

[1] B. Palace, "Data Mining: What is Data Mining?," University of California, Los Angeles, June 1996. [Online]. Available: http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm. [Accessed 11 April 2015].

[2] Tutorials Point, "Data Mining - Classification & Prediction," Tutorials Point, 2015. [Online]. Available: http://www.tutorialspoint.com/data_mining/dm_classification_prediction.htm. [Accessed 11 April 2015].

[3] P.-N. Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, Boston: Pearson Education, Inc., 2006.

[4] The University of Waikato, "Weka 3: Data Mining Software in Java," The University of Waikato, [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Accessed 11 April 2015].

[5] University of Ljubljana, "Data Mining - Fruitful and Fun," University of Ljubljana, 2015. [Online]. Available: http://orange.biolab.si/. [Accessed 11 April 2015].

[6] J. W. Eaton, "GNU Octave," 2013. [Online]. Available: https://www.gnu.org/software/octave/. [Accessed 11 April 2015].

[7] L. van der Maaten, "Matlab Toolbox for Dimensionality Reduction," 2015. [Online]. Available: https://lvdmaaten.github.io/drtoolbox/. [Accessed 11 April 2015].

[8] SciPy Developers, "SciPy.org," Enthought, 2015. [Online]. Available: http://www.scipy.org/. [Accessed 11 April 2015].

[9] I. Jolliffe, Principal Component Analysis, New York: Springer-Verlag Inc., 2002.

[10] L. Breiman and A. Cutler, "Random Forests," University of California, Berkeley, [Online]. Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. [Accessed 11 April 2015].

[11] S. Parkes, "Practical questions on tuning Random Forests," Stack Exchange, Inc., 25 March 2013. [Online]. Available: https://stats.stackexchange.com/questions/53240/practical-questions-on-tuning-random-forests. [Accessed 11 April 2015].

[12] G. Amooee, B. Minaei-Bidgoli and M. Bagheri-Dehnavi, "A Comparison Between Data Mining Prediction Algorithms for Fault Detection (Case study: Ahanpishegan co.)," *IJCSI International Journal of Computer Science,* vol. 8, no. 6, pp. 425-431, 2011.

[13] J. Awwalu, A. Ghazvini and A. Abu Bakar, "Performance Comparison of Data Mining Algorithms: A

Case Study on Car Evaluation Dataset," *International Journal of Computer Trends and Technology (IJCTT),* vol. 13, no. 2, pp. 78-82, 2014.

[14] K. R. Lakshmi, M. Veera Krishna and S. Prem Kumar, "Performance Comparison of Data Mining Techniques for Predicting of Heart Disease Survivability," *International Journal of Scientific and Research Publications,* vol. 3, no. 6, pp. 1-10, 2013.