CSC 869: Data Mining, Spring 2015, SFSU Instructor: Dr. Hui Yang

Term Project Report on:

**INSTRUCTOR ASSESSMENT SYSTEM**

By: Pratik Jaiswal

Problem being addressed: To analyze the opinion of students about their instructors in an educational institution. In this project, the students using data mining will evaluate instructors and further prediction of whether instructors will be invited to faculty classes or not can be done.

This is an important aspect since students are expected to get best instructors in their education system.

Dataset:

Initially, I was using instructor data with attributes like instructor degree, instructor experience, assessment score, degree type and the acceptation. Decision tree classifier was providing certain observations out of this like:

**If ((Assessment score == EXCELLENT) && (Experience == FALSE)) then, ACCEPT the instructor**. This is because even the experience is not that great (NEW instructor); assessment score is excellent that assures education manager to promote the instructor in future semesters.

Later on, I found that this dataset is not sufficient in order to create a robust system since the above-mentioned attributes like instructor degree do not give more insights on how they affect the accept/reject decisions from education managers.

Part I: Instructions on compiling and running the program(s)

Programming Language: Python, Development Environment: Mac OS X

The source code and the results can be found in:

“**Lack of Proper Data Mistake - Files(corrected)”** folder with ***professor.arff*** as a dataset file and it can be directly imported into the WEKA.

The decision tree classifier output is there in the same folder: ***decisiontree.txt*** and the visualized tree is in image: ***DecisionTreeVisualization.png***

Classifier accuracy: **92.85%** (for smaller dataset)

Confusion Matrix:

a b <-- classified as

9 0 | a = yes

1. 4 | b = no

Result: Mistake was a lack of proper data. So, in order *to correct this* mistake, I did more research and landed up on **RateMyProfessors(RPM)** data with more informative attributes like easiness, helpfulness, clarity, overall ratings(value falls between 0 and 5) and total ratings.

Fetching the data from **RPM** was a complicated task since it requires a scrapper to be written and fetch everything from the URL using python. In order to fetch the entire data, run the program ***fetchdata.py***, which is there in **“Dataset Obtain”** folder. This program uses scraperwiki library that gives the data in wiki like fashion and BeatifulSoup library for navigating and copying the data from specified URL (in my case, ratemyprofessor website).

To run fetchdata.py, you need to install libraries scraperwiki and BeautifulSoup using pip and brew like:

(pip should already be installed: [*https://pip.pypa.io/en/latest/installing.html*](https://pip.pypa.io/en/latest/installing.html) & for brew:

*ruby -e "$(curl -fsSL* [*https://raw.github.com/Homebrew/homebrew/go/install*](https://raw.github.com/Homebrew/homebrew/go/install)*)”* – run this ruby command in terminal)

*brew install poppler*

*pip install scraperwiki*

*Linux: sudo apt-get install python3-bs4, Mac: pip install beautifulsoup*

Depending on the platforms, dependability issues can be resolved and above commands should work.

Running instruction: ***python fetchdata.py***

Above program will take a huge amount of time, since it fetches the information of instructors from A-Z series. Therefore, to show an explicit example of how it works, I have written a *small module* that fetches single professor information.

File to fetch single instructor is named ***fetchprofessor.py*** and it is in the same folder **“Dataset Obtain”.**

Running instruction: ***python fetchprofessor.py***

It also gives data in SQLite form, which can be used in SQL program to fire queries.

Furthermore, after obtaining the data, in order to do analysis, I ***have converted the csv data: instructor.csv into arff format: instructor.arff for WEKA.*** A lot of preprocessing came in with handling of missing values, removing the blank entities, etc.

For row removal with values like “Unknown” & “Other”, a program called ***rowremoval.py*** is written in python. File can be located in the folder named **“Row Removal Preprocess”** and it makes use of *instructor.data* as an input that is present in the same folder.

Running instruction: ***python rowremoval.py***

Another preprocessing step was to remove the unwanted blank spaces in the file. This can be achieved using the below terminal command:

***tr -s '[:blank:]' ',' <instructor.data > trim.data***

GUI applications:

In order to validate the results, I have written applications for a computer system Windows/Mac/Linux and Android OS (mobile devices).

The video of an Android application run is at: <https://youtu.be/lNnFkzoiG4s>. The source code for the entire application can be found in “***UI Applications/Android Application”*** folder.

For simply installing application: you can transfer the [.apk](https://github.com/jpratik21/Data-Mining-Term-Project-CSC-869/blob/master/Instructor%20Assessment%20System/bin/RateMyProfessor.apk) file to your Android device. Also, computer application can be found in ***“UI Applications 🡪 Windows/Mac Application”*** folder. The file is named rating.jar and simply double click on it to run the jar file.

***double click rating.jar file***

The source code can be found in the same folder ***“UI Applications 🡪 Windows/Mac Application”.*** Files in the ‘Windows/Mac Application’ folder are:

1. Comment.java 🡪 For handling the comments/reviews on the instructor.
2. Search.java 🡪 For searching the instructor into the dataset
3. rating.sql 🡪 This is a SQLite file that can be used to fire queries if you want to rate a professor.

All above 3 files have dependency on JDBC driver and so, it needs a manual configuration for SQL. ***Please run rating.jar in order to see the GUI application of it.***

Naïve Bayesian Classifier:

I have implemented a Naïve Bayesian Classifier that identifies correctly and incorrectly assigned entities. The program for classifier can be located in folder **“Naive Bayes”** by name classifier.py.

Running instruction: ***python classifier.py instructor.data instructor.test***

For an instance, if my *dataset – instructor.data* has 7490 samples of professors then, my classifier gives 81.06% of accuracy in identifying the correctly classified instances.

=== Confusion Matrix ===

a b <-- classified as

6000 0 | a = accepted

815 675 | b = rejected

Using 10-fold cross validation (WEKA), classifier’s accuracy = 89.07%. The dataset for this is ***instructor.arff***, which is in **“Dataset Obtain”** folder.

------Decision Tree Classifier WEKA:

------R

Part II: Description of the main strategies for the project

Generation of dataset: The strategy of **curling the URL** by developing a **scrapper** was useful since it saved a lot of time and manual work. The integration of **scraperwiki** that gives data in wiki like fashion helped me to create a JSON file of instructor information. Preprocessing was intense to check each column for special characters and this was important to create a compatible arff file for WEKA.

Secondly, missing values were cleverly handled by making an array for suspected words and filtering out those entries in order to remove them. More frequent words were: ***Unknown and Other.*** Similarly, trimming out the blank entries with **“tr”** command on Mac platform.

**BeautifulSoup** library for python made it easy to navigate to different pages from terminal and grab the required data (version bs4)

Classifier construction:

Evaluations & Observations:

Specify for each step, what software did you use/ implemented your own version

Own version: steps to validate the correctness of the program

Part III: evaluation results and discussions

1. Describe/discuss evaluation results
2. What are the effects of data size, min support, min conf
3. Compare classifier with Naïve Bayes Implemented in homework

Part IV: Conclusions

Part V: Comments and suggestions

Future class improvements - inclusion of R

Comments/thoughts/suggestions on projects/class