Analyzing crowd-sourced data about local businesses

Vinay Vasant More
vm5533@rit.edu

Onkar Anil Deorukhkar
oad3132@rit.edu

Pratik Shirish Kulkarni
psk7534@rit.edu

Prasad Girish Chitnis
pgc5277@rit.edu

***Abstract*— Data mining can be used to build classification models which can be used to analyze current trends, trend setters with respect to a particular business. This paper presents results of multiple decision trees for analyzing crowd-sourced data about local businesses obtained from 'YELP Dataset' to predict the quality of the services of a business. Tools used for this project are Weka, R/Rattle.**

I. INTRODUCTION

Different types of businesses are flourishing rapidly and competitively. It is necessary for this businesses to provide better services in order to grow and become better than their competitors. To achieve this, it is important to analyze the crowd-sourced data from current as well as potential new users. For our project, we have selected the 'Yelp' crowd-sourced data for local businesses made available by https://www.yelp.com/dataset_challenge.

This data initially consisted of 3 JSON files for USER, BUSINESS & CHECK-INS data. USER data with 11 attributes and 552339 instances. Each instance in USER data describe a single user information and attributes described the user_id (encrypted user id), review_count (as a continuous value), average_stars (star rating from 1 to 5, rounded to half-stars), votes (as a continuous value), yelping_since (date, formatted like '2012-03'), fans (as a continuous value).

BUSINESS Data with 15 attributes and 77445 instances. Each instance in BUSINESS data describe a single business information and attributes described the business_id (encrypted business id), 5 attributes describing the full localized address, stars (star rating from 1 to 5, rounded to half-stars), review_count (as a continuous value), categories (localized category names), hours (the day of a week with open and close hours), multiple attributes describing availability of the facilities like parking facility, wheel-chair accessibility, take-out, takes-reservation, alcohol, etc.

CHECK-INS Data with 3 attributes and 55569. Each instance in CHECK-INS data describe a single check-in information and attributes described the business_id (encrypted business id), number of check-ins in every hour time-slot for the whole week.

We have two classification results to showcase by analyzing our dataset.

PART I - Predicting a star rating for businesses based on the category, check-ins and review count. Referred as PART I from here onwards.

PART II – Predicting the number of fans for particular user based on number of votes for each vote type, number of reviews written by user and star rating received by a user. Referred as PART II from here onwards.

a. Cleaning phase for PART I:

During the cleaning phase for PART I, for our BUSINESS dataset we did the binning for business categories as initial dataset had over 400 different categories. It was difficult to deal with 400 categories and we knew these can be logically grouped based on the services that they have to offer. Then, we successfully binned it into 5 broad categories Food, Shopping & Entertainment, Services, Health & Fitness and Miscellaneous.

We wrote a JAVA program to convert BUSINESS and CHECKINS JSON dataset into a customized CSV file with required fields and binned value for category field. During this cleaning, we were able to remove 10 attributes from BUSINESS data set and merged number of check-ins by a particular business from CHECKINS dataset into final csv file.

These 10 attributes were not contributing towards the classification or impacting our final model. Six of them were related to full localized address and others were describing the information regarding the additional facilities that they have to offer. There were some businesses with no check-in information available from CHECKINS dataset. We removed these instances marking them as missing or erroneous data.

After cleaning, we had total 6 attributes ("business_id", "name", "category", "stars", "review_count", "checkins") and 55569 instances remaining.

b. Cleaning phase for PART II:

During the cleaning phase for PART II, for our USER dataset we did the binning for 'fans' attribute. The fans attribute specifies number of fans a particular user has. As the value is continuous, it couldn't be used for classification effectively. Binning was performed on the 'fans' attribute to divide the users into two groups, such as, users with less number of fans and users with high number of fans based on the threshold value. In order to classify the data more precisely, a new attributes 'fun' and 'useful' were created using the values from 'votes' and 'review_count' attribute. Number of votes indicate reliability of user whereas review count indicates the activeness of a particular user. This collectively determines the user's usefulness.

We wrote a JAVA program to convert USER JSON dataset into a customized CSV file with required fields. During this cleaning, we were able to remove 4 attributes from USER dataset which were not contributing towards the classification or impacting our final model e.g. friends (the other user_ids which are friends with this user), compliments (cute, funny, plain, writer, note, cool), elite (number of years since the user has been a highly values customer). As the number of instances for USER dataset were over 550000, we have considered only 50000 instances for showcasing classification with available tools.

After cleaning, we had total 9 attributes ("user_id", "yelping_since", "votes", "review_count", "fans", "average_stars", "usefulness", "fun", "popularity") and 50000 instances remaining.

II. Database Design

The database used in this experiment consists of various characteristic information about several businesses and the reviews given to those businesses by their respective users. The original data obtained was in a JSON (javaScript Object Notation) file. It was converted to customized CSV(Comma separated Values) file with cleaned data using a java program.

a. Business table:

After cleaning the BUSINESS dataset we have, following final attributes in the Business table ("business_id", "name", "category", "stars", "review_count"). business_id is the primary key to uniquely identify each business.

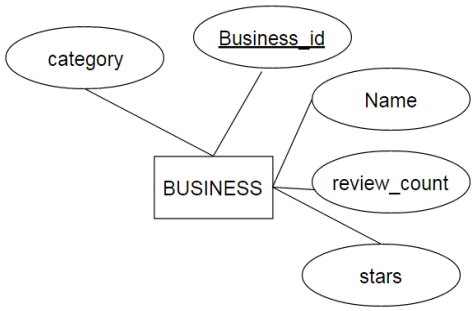

Table shows BUSINESS dataset with business_id as primary key.

b. Check-ins table:

After cleaning the Check-ins dataset we have, following final attributes in the Check-ins table ("checkin_id", "b_id", "number of checkins"). checkin_id is the primary key to uniquely identify each check-in by a user however, b_id is the foreign key which connects the business to a particular check-in.

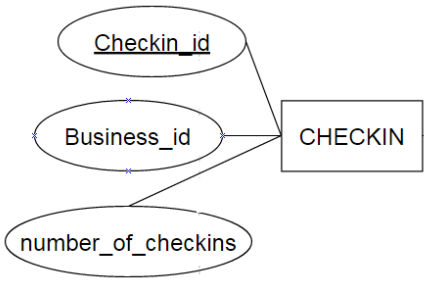

Table shows Check-ins dataset with Checkin_id as primary key.

c. User table:

After cleaning the User dataset we have, following final attributes in the User table ("user_id", "votes", "fans", "average_stars", "review_count", "yelping_since", "fun", "usefulness", "popularity"). user_id is the primary key to uniquely identify each user.

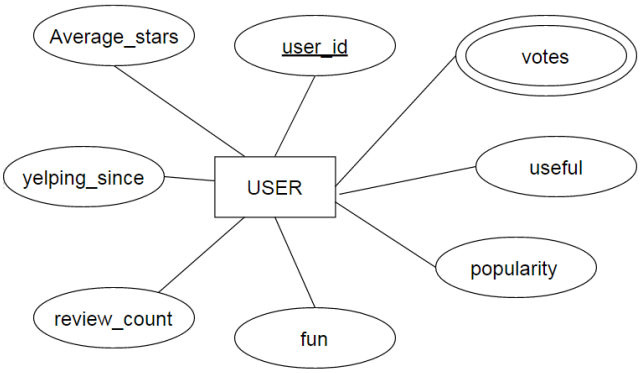

Table shows USER dataset with user_id as primary key.

III. Analysis

a. PART I - Predicting a star rating for businesses based on the category, check-ins and review count.

The goal here was to classify stars (1 stars, 1.5 stars, 2 stars, 2.5 stars, 3 stars, 3.5 stars, 4 stars, 4.5 stars) such that a decision tree can be formed that will be able to predict which star rating a business is likely to get considering the category, number of checkins and review_count received by users. Our hypothesis involved classification so we initially decided to use a NaiveBayes Algorithm. We initially selected this classification algorithms because we wanted a concept like

'probability' to classify our data by handling multiple evidence. This classifier is based on the Bayes rule of conditional probability. NaiveBayes algorithm uncouples multiple pieces of evidence, and to treat each of piece of evidence as independent. So, it helps in multi class prediction as well. A Naive Bayes classifier performs better compare to other models like logistic regression when independent pieces are to be handled and also you need less training data.

When we built the NaiveBayes model, we noticed that the correctly Classified Instances were 73.64%. This made us suspicious that frequencies of data, term weighting methods and extra work that NaiveBayes does to handle multiple evidences are affecting the accuracy. J48 classification method has better performance than NaiveBayes in terms of frequency and term weighting methods. When we built the J48 model, we saw the correctly classified instances were 83.65.

b. PART II – Predicting the number of fans for particular user based on number of votes for each vote type, number of reviews written by user and star rating received by a user.

The goal here was to classify the users such that a decision tree can be formed that will be able to predict whether a particular user will have high or low number of fans. To reduce the cost of the decision making process, it was necessary to create a decision tree with lesser number of nodes which is able to predict accurate decisions. This was achieved by classifying the data using less number of attributes. The attributes not considered for classification were votes and user_id. Our hypothesis involved classification so we initially decided to use a J48 decision tree algorithm as it is very efficient at handling continuous and discrete attributes. So, first we performed classification using Weka over all the attributes of user using J48 algorithm. The overall accuracy of the resulting confusion matrix was 77.3501%. The decision tree formed had several number of nodes which implies a higher cost for predictions using the decision tree.

Random-Tree selects the most optimal tree from a set of all possible trees while J48 builds the tree by recursively partitioning the data set. Therefore, in our case, Random-Tree provided better classification model as the dataset consisted highly dispersed values i.e. the differences between two consecutive values were too high in many cases which leads to less effective partitioning by J48 algorithm. Random-Tree algorithm in Weka, resulted into classification with 83.0713% accuracy.

IV. RESULTS

For PART I Classification Hypothesis the accuracy of NaiveBayes algorithm was found to be about 73.64% . After Further research we decided to use J48 as we thought it was a better fit to the given hypothesis. J48 improved the accuracy to 83.65% .

=== Confusion Matrix by J48===

a	b	c	d	e	<-- classified as
12607	655	479	413	360	| a = 2 stars
514	**12157**	264	257	490	| b = 3 stars
821	804	**9153**	507	369	| c = 1 stars
712	701	556	**8256**	249	| d = 4 stars
307	330	143	151	**4314**	| e = 5 stars

This confusion matrix shows the performance of J48 classification model. The diagonal values represent correctly classified instances however, the non-diagonal value are incorrectly classified instances or errors. As we have,

For PART II Classification Hypothesis the accuracy with J48 algorithm was found to be 77%. We then used Random- tree algorithm for the user's dataset and generated the most optimal decision tree and classified the data with 83% accuracy with binning.

=== Confusion Matrix ===

a	b	<-- classified as
6655	6993	| a = high
2094	37936	| b = low

This confusion matrix shows that 44591 out of 53678 instances of the data were predicted accurately i.e. 83% accuracy.

V. LESSONS LEARNED

Binning: For dataset with multiple variables having continuous values it is beneficial to select binning operation on that dataset as it helps us achieving precise and accurate decision tree.

Decision Trees: Decision-trees are formed by recursively partitioning the training data set. Every node represents a query over an attribute's value. A path to the child node is created using this value using this value. Every time a new node is added to the decision tree, a subset of the training data set is used to determine the logical test leading to that node. The logical test leading to the root of the node is determined based on all the elements in the data set [4].

When to use NaiveBayes Algorithm?
This classifier is based on the Bayes rule of conditional probability. NaiveBayes algorithm uncouples multiple pieces of evidence, and to treat each of piece of evidence as independent. So, it helps in multi class prediction as well. A Naive Bayes classifier performs better compare to other models like logistic regression when independent pieces are to be handled and also you need less training data.

When to use Random-Tree algorithm?

Random-Tree selects the most optimal tree from a set of all possible trees while J48 builds the tree by recursively partitioning the data set. Therefore, in our case, Random-Tree provided better classification model as the dataset consisted highly dispersed values i.e. the differences between two consecutive values were too high in many cases which leads to less effective partitioning by J48 algorithm.

REFERENCES

[1] Zheng~Yao,Peng~Liu,Lei~Lei,Junjie~Yin,"RX4.5 Decision Tree Model and Its Applications to Health Care Dataset",International Conference on Services Systems and Services Management, 2005.

[2] Amany Abdelhalim, Issa Traore,"A New Method for Learning Decision Trees from Rules",International Conference on Machine Learning and Applications, 2009.

[3] Wirot Yotsawat, Anongnart Srivihok,"Inbound tourists segmentation with combined algorithms using k-means and decision tree",10th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2013.

[4] MD. Ridwan Al Iqbal,Saiedur Rahman,Syed Irfan Nabil,Ijaz Ul Amin Chowdhury,"Knowledge Based Decision Tree Construction with Feature ImportanceDomainKnowledge",7th International Conference on Electrical and Computer Engineering, 2012.

[5] Hang Yang, Simon Fong,"Optimized Very Fast Decision Tree with Balanced Classification Accuracy and Compact Tree Size",IEEE Conference Publications, 2011.

[6] Dr. Md. Ali Hussain1, Dr. M. Kameswara Rao, Dr. Ali Mirza Mahmood,"An Optimized Approach To Generate Simplified Decision Trees",IEEE Conference Publications, 2013.