Lukasz Borowczak (lboro2), Tony Wu (tonywu3)
CS412

Project report

Introduction

In this project, we aimed to predict how a user will rate a particular movie based on previous users' ratings, their profile, and the movie information. The most challenging part of this project was cleaning the data and shaping it to a usable format. We used various techniques for dealing with missing data including: applying the most common value, using the average value, and treating missing values as their own value. In this project, we will be using a decision tree for our classifier and gini as our parameter to split the different attributes. The significance of this project can be extended beyond the scope of movie ratings. Large companies today are already using classification techniques to selectively push advertisements to individuals to boost their sales. This project will allow us to gain some insight to how that process might work.

Related work

https://pdfs.semanticscholar.org/e286/5994c80238daa302b4960b6274ca2e145104.pdf

In the data files, there are many missing data values for attributes like Age, Occupation, and Gender for which we needed a way to deal with the missing data. To gain insight on which method we should use to handle the missing data, we looked at a paper by Jerzy W. Gryzmala-Busse and Ming Hu from the University of Kansas, "A Comparison of Several Approaches to Missing Attribute Values in Data Mining". In their paper, they test 9 different approaches to handling missing data on various datasets and reports their error rates using 2 different classifiers.

From the paper, we decided to use "most common attribute value" for missing categorical data like Gender. We also used the method of "treating missing attribute values as special values" for the Occupation attribute. We chose these methods due to their simplicity of implementation. The drawback of these methods is that it may not be a good way of representing missing values according to our particular dataset.

http://www.iaeng.org/publication/IMECS2009/IMECS2009_pp727-731.pdf

In this paper, "Survey of Classification Technique in Data Mining" by Thair Nu Phyu, he examines the various classification techniques (Decision Tree, Bayes) that we discussed in class more deeply. The section on decision trees in the paper talks about the various ways we could split our data on: entropy, gini, or C4.5. The paper gave us a high-level overview of the decision tree algorithm which will be extremely useful when we implement our own algorithm. The paper also reviews Bayes and K-Nearest Neighbor algorithms and best use case scenarios which will help us decide on a specific classifier based on our particular dataset.

Implementation

For our implementation, we decided to use a decision tree classifier from the scikit-learn package. We also used other packages such as numpy and collections to primarily clean up the data and fill in missing values.

For missing gender, we decided to calculate and then use the most common gender, which was Male for the provided data set, and we then transformed it to a binary variable with 0 standing for one gender and 1 standing for the other gender.

For missing age, we calculated the average age for all users, and replaced all missing values with the closest integer age to the average. We applied the same procedure for movies with missing year values, using the average integer year.

For missing occupation, we decided to treat missing the missing data as their own attribute, so we assigned a value of "-1" for those users, as we assumed a missing value meant that they were unemployed.

For movie genre, we implemented one hot encoding which turned every movie genre into its own category. Given a movie, if its genre was the same as the the movie genre in the list, it returned a "1", if it wasn't, it returns a "0". The purpose of this process was to turn categorical data into a format more suitable for classification.

To test our classifier's accuracy, we used one iteration of cross-validation, and ran the program multiple times to get an average value for the accuracy of the model.

Result Interpretation

Our classifier resulted in a score of .37737 on Kaggle against the validation set. Meaning our classifier correctly predicted about 38% of the validation data. Interestingly, using the entropy parameter versus gini index resulted in a nominal score difference in our testing. For entropy, we got a score of 0.398 and for gini index, we got a score of 0.400. This suggests that our data is consistent across different impurity measures. To improve our score, we can test out other classifiers or use different measures to handle missing data values in a more representative way. We can also try tweaking the specifics of the decision tree, but this may be a time-intensive process that may not result in much improvement. By adjusting the max depth and minimum samples per leaf and trying many different values for each, we only managed to raise the classification score less than 0.100, or 10%. An interesting result we found was that having only 13 total nodes in the tree, including leaf nodes, was more accurate at prediction (about 35%) than if we let the tree grow without limits or with the default limits(about 31%), but still less accurate than a more complex decision tree with more nodes.

Work Distribution

Lukasz: 50%, did all of the code, worked on some of the report.

Tony: 50%, worked on the report, found relevant research papers, found examples of decision tree classifier usage.