All’s Fair in Love and Scores

MSCI 446: Data Mining and Warehousing

Lukasz Golab

Wednesday November 30th, 2016

Maisie Brown

Connor O’Brien

Dale Drasnin

Patricia Hall

**Table of Contents**

Abstract 1

Introduction 2

Related Work 2

Data 2

Results 3

Naive Bayes 5

Decision Trees 5

K-Nearest Neighbor 7

Linear Regression 7

Multi-Regression 9

Logistic Regression 9

Association 10

Clustering 10

Improvements 13

Conclusions 14

References 15

Appendices 16

Appendix A - Survey 16

Appendix B - Code 21

B-1 Naive Bayes 21

B-2 Decision Trees 23

B-3 K-Nearest Neighbor 25

B-4 Linear Regression 26

B-5 Multi-Regression 28

B-6 Logistic Regression 31

B-7 Association 33

B-8 Clustering 36

**Table of Figures**

Figure 1: Frequency of Grade Change Categories 3

Figure 2: Frequency of Program 3

Figure 3: Frequency of Age 4

Figure 4: Confusion Matrix for Naive Bayes 5

Figure 5: Confusion Matrix of Decision Tree A 6

Figure 6: Confusion Matrix of Decision Tree B 7

Figure 7: Linear Regression Model for Age and Grade Change 8

Figure 8: Mean Residual Sum of Squares 8

Figure 9: Sum of Distances of Data Samples to their Closest Cluster Center vs. the Values K 11

**Table of Tables**

Table 1: Effect on Grades for Different Genders 4

Table 2: Effect on Grades for Different Relationship Start Times 4

Table 3: Sample Input Data for K-Nearest Neighbors 7

Table 4: Sample Input for Linear Regression using Age and Grade Change 8

Table 5: Sample Input for Linear Regression using Start Year and Grade Change 8

Table 6: Sample Input for Linear Regression using Age and Grade Change 8

Table 7: Input Data Columns for Multi-Regression 9

Table 8: Sample Input Table for Association Rules 10

Table 9: Optimal Clusters with K=10 12

Table 10: Rank of Coefficients for Logistic Regression Model 14

**Abstract**

In this paper, we perform a data mining study to analyze the effect of entering a relationship on university students’ academic performance. We created a survey based on explanatory variables that we wanted to analyze. We applied classification, prediction, association rules and clustering to the collected data. Our results reveal interesting relationships between the frequency partners visit each other, the academic year the relationship started, the gender and their grade change.

**Introduction**

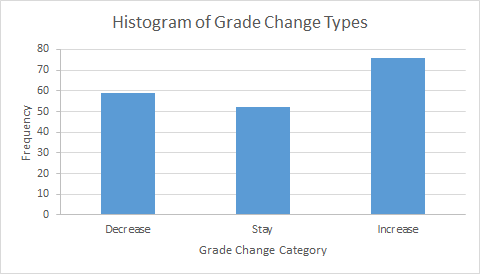
The purpose of our project is to determine whether there is a correlation between entering a romantic relationship and academic achievements. Our main hypothesis is that students’ grades will decrease when they enter a romantic relationship because less of their time is dedicated to schoolwork. This problem is interesting because there is little documentation on the effect of a romantic relationship and academic achievements in university. Results from this problem can be used by future students, academic advisors and guidance counselors by addressing what changes in academic achievement to expect when entering a relationship. The main results found are that male students who enter a relationship in first year will see their grades decrease.  
  
**Related Work**  
  
There is some previous work with similar data. The Department of Counseling Psychology at Santa Clara University published a study about academic and various other correlates of adolescent dating. This study is different from our analysis because it is studying Grades 8, 10 and 12. The results from this study show that adolescents who date frequently exhibit lower grades in school. There is a Social Science Research journal titled “Adolescent academic achievement and romantic relationships” that studies the link between academic achievement and romantic involvement during the adolescent period. The study takes into account the individual's orientation toward school, the individual’s familial relationship and the individual's demographic and shows that romantic partners’ grades are related to the individuals’ grades. Using this knowledge, the study is able to predict individual’s grades based on their romantic partner’s grades. We found one journal published by the College Student Journal titled “Breakup Effects on University Students’ Perceived Academic Performance.” This journal explores university student’s perceived academic performance as a result of relationship breakups. The results from this study show that breakups negatively affect students’ perceived academic performance. This study is different from our analysis because it focuses on students’ perceived academic performance and breakups whereas our analysis focuses on actual academic performance and entering a relationship.   
  
**Data**  
  
The data is collected using a survey. The survey is created by brainstorming potential explanatory variables then creating corresponding questions for those variables. The survey begins with two questions:

1. Are you in a relationship and in university?
2. Have you ever been in a relationship while in university?

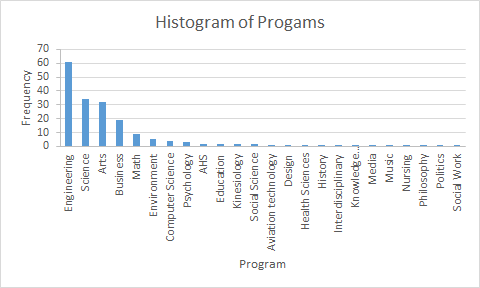
The survey proceeds if the answer to at least one of these questions is yes because we are interested in looking at the grade change of people whose relationship status has changed during university. Some data cleaning is required. Data points where the answers to both questions above are “no” are omitted. Data points with grades on a 4-point scale are converted to a 100-point scale using the respective school’s conversion chart. Data points missing fields such as “grade average for the term before relationship started,” and “grade average for the term the relationship started” are omitted because the grade change cannot be calculated. “Grade change” is calculated by subtracting the average for the term before the relationship started from the average for the term the relationship started. The numbers are rounded to the nearest whole number.

Preliminary information about the data such as frequencies of grade change types, programs and ages are collected by counting the occurrence of each category. Preliminary analysis between explanatory variables and the class variable such as gender and grade change, and start year and grade change are collected by counting the occurrence of each sub-category and displaying it as a percentage of the category. For example, one sub-category is “females with a grade increase.” This represents 43% of the “female” category.

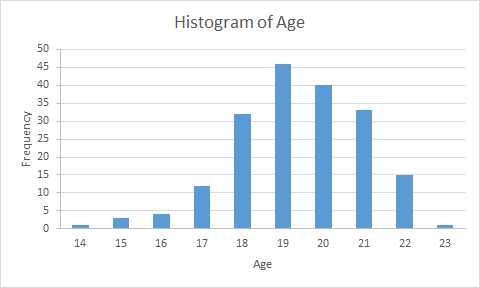
*Figure 1: Frequency of Grade Change Categories*



*Figure 2: Frequency of Program*



*Figure 3: Frequency of Age*



*Table 1: Effect on Grades for Different Genders*

|  |  |  |  |
| --- | --- | --- | --- |
| **Gender** | **Decrease** | **Stay** | **Increase** |
| Female | 27% | 30% | 43% |
| Male | 41% | 22% | 36% |

*Table 2: Effect on Grades for Different Relationship Start Times*

|  |  |  |  |
| --- | --- | --- | --- |
| **Relationship Start** | **Decrease** | **Stay** | **Increase** |
| Before university | 23% | 42% | 35% |
| 1st year | 46% | 29% | 24% |
| 2nd year | 30% | 23% | 47% |
| 3rd year | 33% | 30% | 37% |
| 4th year | 70% | 15% | 15% |

**Results**

In this section we summarize our findings of each algorithm we perform on our dataset, including the purpose, motivation, hypothesis and result for each instance.

**Naive Bayes**  
The purpose of Naive Bayes is to find the probability of a the class variable of grade change given a set of independent explanatory variables. Naive Bayes is classified as “Naive” because it assumes the explanatory variables are independent. The motivation behind using this model is to see if there is a grade change when entering a relationship given that a set of various explanatory variables also exists. We binned the class variable into 5 categories: grade increase greater than 4%, grade change 0 to 4%, no change, grade change 0 to -4% and grade change less than -4%. These bins are chosen arbitrarily by observing the data and creating sections of equal sizes. All of the explanatory variables are one-hot encoded in order to avoid the implicit bias that a higher numerical value is better than a lower numerical value. The input table consists of all 10 explanatory variables: School, Program, Religious, Gender, Start Year, Age, Partner Age, Hour Away, Live Together and Frequency. The algorithm has an accuracy of 57% and a cross validation using K=5 of 28%. The confusion matrix shown below.

*Figure 4: Confusion Matrix for Naive Bayes*

[[30 9 4 2 0]  
 [12 29 9 1 1]  
 [ 4 4 20 2 2]  
 [10 3 1 16 1]  
 [ 6 3 3 3 12]]

The confusion matrix shows the predicted vs actual values for ['Greater than 4%', 'No change', 'Less than -4%', '0 to 4%', '-4% to 0']. The low accuracy of this model shown by the confusion matrix and the discrepancy between the 5-fold cross validation accuracy and the model accuracy indicate overfitting. This could mean that the variables are not independent or that we do not have a large enough sample size and therefore Naive Bayes is not considered for further analysis.

**Decision Trees**

The purpose of using decision trees is to break down the dataset into smaller and smaller subsets to build classification models. Two different decision trees are created for this report and can be seen in Appendix B-2. Decision tree A uses input table A containing all the explanatory variables and decision tree B uses input table B, containing a select few explanatory variables. Input table B is used to compare the original hypotheses with the decision tree algorithm outputs to determine if there is any validity to the originals. When using input table A, the accuracy of the model is 39.7%. The accuracy of the model when input table B is used is 37.4%. The accuracy of both models is very poor. This is likely caused by the small sample size and overlapping individual types.

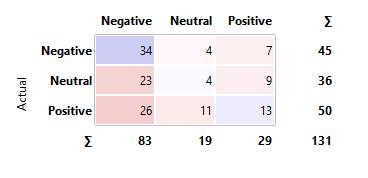
The input data includes both categorical and numerical data. The target variable is bin\_grade\_change which is a binned variable of grade\_change. Three bins are used; positive, negative, and neutral. Positive grade change includes data where the grade increases upon entering a relationship. Negative grade change includes data where the grade change is negative upon entering a relationship. Neutral includes data where no change in grade is seen upon entering a relationship.

The original hypotheses are made to hypothesize the grade change of different types of people. The general hypothesis is if an individual is above the age of 19, their grades will increase upon entering a relationship. Several specific hypotheses are made to suit a number of different types of people. If an individual is below 20 years old and their frequency of visits per month is greater than 25, they will see a decrease in grades upon entering a relationship. An alternate hypothesis that contrasts genders is that if an individual is female and below 20 years of age, their grades will decrease upon entering a relationship whereas if the individual is male and below the age of 20, they will see no change in their grades upon entering the relationship.

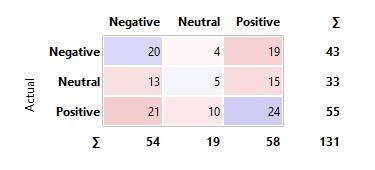
Using input table A in Appendix B-2, the decision tree created has partner age as the root node and the most successful (percentage wise) branch hypothesis is “IF Partner\_Age <19, Program = Engineering, Start Year > 0, and Age < 19 THEN grade will decrease upon entering a relationship.” To compare with the hypotheses made, some variables are ignored and input data B in Appendix B-2 is used. The most interesting two hypotheses resulting from this decision tree are “IF age > 19, frequency > 3, and gender = female THEN grades will increase upon entering a relationship” with 49.2% accuracy and “IF age > 19, frequency > 3, and gender = male THEN grades will increase upon entering a relationship” with 53.3% accuracy. This is an interesting result because both agree with the initial hypotheses “IF female and age<20 THEN grades will decrease upon entering a relationship” and “IF male and age<20 THEN grades will remain constant or increase upon entering a relationship” with the added constraint of having frequency be greater than 3.

Both confusion matrices below display large errors for positive, neutral, and negative grade change. These large errors in combination with the low accuracy scores show the inadequacy of using decision trees on this data. The most likely cause is that the different types of individuals are not able to be differentiated by the decision tree algorithm without building too large of a tree. Therefore, to increase accuracy, we should seek out more specific types of people and increase sample size.

*Figure 5: Confusion Matrix of Decision Tree A*



*Figure 6: Confusion Matrix of Decision Tree B*



**K-Nearest Neighbors**

The purpose of doing K-Nearest Neighbors is to classify variables based on a similarity measure. Our motivation behind using K-Nearest Neighbors is to use an algorithm that does not train the data and compare the results with algorithms that do train the data. The class variable is converted to a binary variable by using “grade\_increase” where 1 represents “yes” and 0 represents “no” (decrease or no change). Since the class variable is binary, an odd number of K is used to ensure that there is always a majority or tie-breaker. We hypothesize that the majority (>50%) of the observations will be classified correctly based on the 3 nearest neighbors.

*Table 3: Sample Input Data for K-Nearest Neighbors*

|  |  |  |
| --- | --- | --- |
| frequency | start\_year | grade\_increase |
| 31 | 4 | 0 |

The accuracy score for this model using 3 nearest neighbors is 73%. We use a value of K=10 for cross-validation. This value is chosen by researching the best values for K and choosing the most popular one. A lower value of K means less variance and more bias, therefore a higher value of K is chosen. Based on the cross validation value of 37%, we conclude that K-Nearest Neighbors is overfitting the data. This is most likely because the algorithm only looks at the nearest immediate neighbors therefore could be fitting to an outlier in the data. This can be mitigated if we have more data to test because there will be more smoothing.

**Simple Linear Regression**

The purpose of doing linear regression is to model the relationship between one of the explanatory variables and our class variable of grade change. We attempt to model the relationship with age and the class variable, with start year and the class variable and with frequency of visits and the class variable. We choose these explanatory variables because we expect them to correlate with grade change the most. We hypothesize that as you get older and more mature, your grades will not change as much when you enter a relationship. Similarly, we hypothesize that the later the start year is, the less the grades will change because you are more familiar with study habits and how to achieve higher grades. We hypothesize that the more frequently you visit your significant other, the more your grades will decrease because less of your time is able to be spent on studying. The data we input for each linear regression is shown below.

*Table 4: Sample Input for Linear Regression using Age and Grade Change*

|  |  |
| --- | --- |
| Age | Grade\_Change |
| 18 | -14 |

*Table 5: Sample Input for Linear Regression using Start Year and Grade Change*

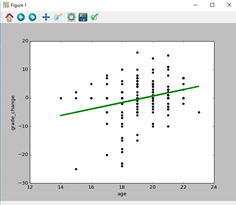
|  |  |
| --- | --- |
| Start\_Year | Grade\_Change |
| 1 | -14 |

*Table 6: Sample Input for Linear Regression using Age and Grade Change*

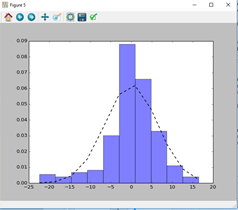
|  |  |
| --- | --- |
| Frequency | Grade\_Change |
| 31 | -14 |

The highest R2 value is 8% and corresponds to age and grade change. The mean residual sum of squares is 40.78 and is normally distributed around zero.

*Figure 7: Linear Regression Model for Age and Grade Change*



*Figure 8: Mean Residual Sum of Squares*



Frequency

Residual Error

This indicates that the linear model generated fits 8% of the data. This is a poor result and indicates that either age does not have a significant impact on grade change when entering a relationship or that the class variable may depend on several explanatory variables. Based on this, we decide to include multi-regression with several of our explanatory variables.

**Multi-Regression**

The purpose of multi-regression is to predict our class variable of grade change using several explanatory variables. Categorical variables are converted into numeric categories in order to be used in the model. We one-hot encode to convert our explanatory variables into numerical variables so that each feature has its own weight. These variables are then concatenated into a data set to be used in our model.

After one-hot encoding our variables, the input dataset is composed of 92 columns, with each column representing a binary variable for each distinct value of the explanatory variable. All but one of the explanatory variables are hot encoded, with Frequency being the one exception The variables that are included are shown below:

*Table 7: Input Data Columns for Multi-Regression*

|  |  |
| --- | --- |
| **Variable Name** | **Number of Columns** |
| School | 28 |
| Program | 25 |
| Religion | 2 |
| Gender | 3 |
| School year when relationship started | 5 |
| Age when relationship started | 10 |
| Partner’s age when relationship started | 15 |
| Long Distance | 2 |
| Live Together | 2 |
| Frequency of Visits per month | 1 |

Depending on the order that these variables are concatenated in, it is found that the R2 value (goodness of fit) is affected. To find the best combination of explanatory variables, a function is written that tests all possible combinations of the 9 explanatory variables and returns the combination that gives the highest R2 value, or the highest accuracy. The function gives us a maximum R2 value of 66.6%, which indicates that the multiple-regression model generated fits 66.6% of our data. This is a good result but potentially misleading as the model may overfit the data.

Multi-regression has a high potential for overfitting. Overfitting in regression models occurs when we try to predict many parameters with a sample data set that is too small. In our model, we have a sample size of 187. Our multi-regression model yields the highest accuracy when we have 92 one-hot encoded variables in our model instead of categorical variables, which leads us to suspect that it overfits the model greatly. In general, we decide that regression is not a good class of model to use for this data set because all but one of our variables are categorical and are better suited for classification algorithms.

**Logistic Regression**

The purpose of using logistic regression is to find the probability of a binary class variable given the value of several numeric explanatory variables. To apply logistic regression to our data, we first convert our class variable (% grade change) into the binary variable “GradesDecrease” which is true when a student's grades go down after entering a relationship and false if grades increase or if grades stay the same. The other variables are one-hot encoded to convert them into numeric data for use in this model. The input table is identical to that used for the multi-regression model (92 binary variables and one for Frequency) but the class variable is different because it is now binary.

We write an algorithm to iterate through all combinations of two features while recording both the model accuracy and the mean of 5-fold validation. We have a threshold of a 5% difference between the two values where any combination with a difference greater than 5% is disregarded as the model is considered to be overfitting to the data. The combination of two feature variables with the highest accuracy that met the difference threshold is Frequency and Start Year with an accuracy of 70.59% and a 5-fold mean accuracy of 69%. We then add a third variable from the remaining 8 and get a combination of Frequency, Start Year and Gender with an accuracy of 73.26% and a 5-fold mean accuracy of 69%. If we try to add more variables, we find that the accuracy does not increase or the 5-fold accuracy goes down, meaning the combination of 3 feature variables is the best result. Logistic Regression is hereafter considered our main model because it gives the highest accuracy without overfitting.

**Association**

The purpose of association rule mining is to find frequent co-occurring associations among a collection of categorical variables. Our motivation is to find which of the categorical explanatory variables are associated with others, including the class variable. The input data is organized as shown below.

*Table 8: Sample Input Table for Association Rules*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **School** | **Program** | **Religious** | **Gender** | **Start\_Year** | **Grade\_Change** | **Age** | **Partner Age** | **Hour\_Away** | **Live\_Together** | **Frequency** |
| University of Western Ontario | Arts | Not Religious | Female | Year 4 | -4% | Age 21 | P\_Age 20 | Short Distance | Separate Housing | 31 days per month |
| Humber College | Business | Religious | Female | Year 1 | -5% | Age 19 | P\_Age 20 | Short Distance | Separate Housing | 31 days per month |

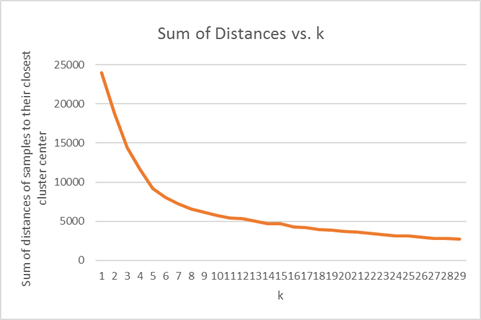
We initially use the suggested minimum support and confidence of 0.01 and 0.9 respectively. This takes too long to run as there are so many individual itemsets that meet the support requirement, so the minimum support is changed to 0.1. After running the Apriori algorithm with minimum support and confidence of 0.1 and 0.9 respectively, there are many rules but the majority of them predict an insignificant variable, such as Live Together or Long Distance. In order to find association rules that give an itemset that includes our class variable Grade Change, we have to lower the minimum confidence to 0.5. Since Apriori does not predict our class variable with a high accuracy or give any association rules that yield significant results, we will not proceed with further analysis.

**K-Means Clustering**

The purpose of clustering is to group a set of data into K optimal clusters, where the points in each cluster are similar to each other but not similar to the points in other clusters. Using K-means, we can group similar types of people together. The motivation behind using this model is to determine if someone’s grade will increase a lot, somewhat increase, stay the same, somewhat decrease or decrease a lot depending on what cluster the person is a part of. Before running K-means on the data collected, we predict that there will be clusters of Engineering students and Computer Science students who attend the University of Waterloo because this was the demographic the survey was predominantly shared with. We also predict that individuals who live with their partner will see their partners more frequently than individuals that don’t live with their partner.

The input table includes all numerical and binary data with the explanatory variables including School, Program, Religious, Gender, Start Year, Age, Partner Age, Hour Away, Live Together and Frequency. We do not one-hot encode the explanatory variables because cluster analysis is the task of grouping a set of data so that points in the same cluster are more similar to each other than to those in other groups. We are able to do this by converting our data from categorical to numerical and do not necessarily need to use one-hot encoding. K-means is run with multiple values of K ranging from K=2 to K=29. The findings are then plotted on the graph below.

*Figure 9: Sum of Distances of Data Samples to their Closest Cluster Center vs. the Values K*



From plotting the total distance vs. K, we determine that the optimal value of K is 10 because this is where the elbow of the relationship is (the sum of distances between each data point and their closest cluster center is 5788.7 when K=10). The data used to plot this graph are shown in Appendix B. The table below shows cluster results for our explanatory variables: School, Program, Religious, Gender, Start Year, Age, Partner Age, Hour Away, Live Together and Frequency.

*Table 9: Optimal Clusters with K=10*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cluster | School | Program | Religious | Gender | Start Year | Age | Partner Age | Hour Apart? | Live together? | Frequency/month |
| 1 | OCAD | Arts | No | female | 2 | 19 | 21 | No | Yes | 6 |
| 2 | Western | Arts | No | female | 2 | 19 | 20 | No | Yes | 4 |
| 3 | Vermont | Politics | No | female | 2 | 19 | 19 | Yes | Yes | 28 |
| 4 | Waterloo | Computer Science | No | female | 2 | 20 | 21 | Yes | Yes | 29 |
| 5 | Vermont | Politics | No | female | 2 | 19 | 19 | No | Yes | 2 |
| 6 | McGill | Business | No | female | 1 | 19 | 19 | Yes | Yes | 29 |
| 7 | Western | Design | No | female | 2 | 19 | 20 | Yes | Yes | 16 |
| 8 | Queens | Psychology | No | female | 2 | 19 | 20 | Yes | Yes | 8 |
| 9 | Waterloo | Engineering | No | female | 2 | 20 | 20 | No | Yes | 4 |
| 10 | U of T | Psychology | No | female | 2 | 20 | 20 | Yes | Yes | 18 |

From observing the 10 clusters we are not surprised that the age of the individual being either 19 or 20 years old while the start year is second year. Most students are either 19 or 20 years old when they are in the second year of university. Also, when comparing the program vs. the school of the various clusters, the results are as expected. For example, the two clusters whose school is Waterloo, the two programs are Computer Science and Engineering. This makes sense because most of the individuals who completed the survey from Waterloo are in Computer Science or Engineering.

We are surprised that for 9/10 of the clusters, the start year of the relationship is in second year. Our group expected to find more clusters where the start year was first year because from our previous experience, more relationships began in first year than second year. Another set of variables our group finds abnormal is that some clusters whose individuals live one hour apart, live together. In the real world this cannot happen because when people live together, they are less than 1 hour away from each other. Also, we expect the frequency to be high (>15 visits/month) if the individual lives less than one hour away from their partner and the frequency to be low (<15 visits/month) if the individual lives over one hour away from their partner. The optimal clusters do not reflect this prediction, as 9/10 clusters do not reflect this prediction.

**Improvements**

After running our algorithms, we find inconsistent results and many instances of overfitting. Based on this we can come up with several improvements to our data that may result in higher accuracy and less overfitting. One of our main issues is that the survey we created does not capture all the results we were expecting to collect. One third of the data initially collected was deleted out of the data set because important fields were missing or the responses did not make sense. This is a sign of there being many outliers in the dataset and ambiguity in the questions. For example, the purpose of the question “How many times on average do you see your partner per month?” was meant to capture the amount of time the individual spent with their partner, however the question could be interpreted in many different ways (an individual may answer 60 because they see their partner twice per day, every day and another person may answer 0.5 because they see their partner 50% of the time). This question should have been worded so that there was no ambiguity in what the question was looking for. It could have been reworded to “How many days per month do you see your partner? (Integer out of 30 days per month)”.

The data collected does not yield conclusive results using data mining rules with high accuracy rates. This may be because the data is not normalized. If someone’s grade changes after they enter a relationship, the change in relationship status may not be the only factor that affected the change in grade. Other factors that affect someone’s grade could include: if the term they were in was more difficult than the previous term or if they had personal issues which affect their class attendance and study habits.

**Conclusions**

Logistic regression is selected as our main model as it has the highest model accuracy and mean accuracy from 5-fold validation.

After running through several test cases, we find that the biggest factor on grade change is the academic year your relationship starts. We find that male students who enter a relationship in first year will see their grades decrease. We find that female students who enter a relationship in first year will not see their grades decrease. This makes sense because these coefficients are the ones with the most positive rank.

*Table 10: Rank of Coefficients for Logistic Regression Model*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Frequency | Year 0 | **Year 1** | Year 2 | Year 3 | Year 4 | **Male** | Female |
| 0.00966098 | -0.35429912 | **0.50396022** | -0.09223246 | -0.00337779 | -0.57571577 | **0.05837089** | -0.58003581 |

We choose the class of this model because our feature variables are all categorical except frequency. We choose this particular subset of classification models because it gives the highest accuracy with the least amount of overfitting.

One lesson learned from this project is that the methods utilized to collect data greatly affect the quality of the conclusions drawn from the analysis. The data for this project was collected using an online survey which returned a large portion of unusable data and many outliers. Some of the questions asked were ambiguous and therefore the responses made it difficult to analyze and draw conclusions from. The most important lessons that our team is taking away from this project are to collect data using more clear and direct questions and to collect a larger sample size. Doing so will likely reduce overfitting, allow the dataset to be split into a larger quantity of more diverse sample datasets to be used for analysis, and reduce the effect outliers will have on the analysis and conclusions.

Various software packages are used when implementing the different data mining algorithms. Orange is a software that is very simple to use for initial data analysis. The data set was uploaded as a csv file, and then we are able to “drag and drop” the algorithms we want to run through the data. Orange then outputs the results of the algorithm but we are not able to trace through the code. Python was used to perform a deeper analysis of the data because we were able to tell the program exactly what we needed outputted and we were able to trace what the code was doing.

Our results impact students in university that are considering entering a relationship or are currently in a relationship. The results of this study can be utilized by students, parents, and academic advisors to help guide students on making important decisions regarding entering a relationship during their university career. Using our algorithms, we could have students respond to our online survey and run analytics based on their data to return a probability of having their grades increase, decrease, or remain the same. The broad impact of our results could be that students who are determined to be at risk of seeing a decrease in grades might postpone entering a relationship until they enter a different category who is more likely to increase grades or stay the same. On a large scale, if a university implements this they may see grades increase and a decrease in first year failure rates.

**References**

<http://stats.stackexchange.com/questions/187595/clustering-with-categorical-and-numeric-data>

<http://datascienceguide.github.io/>

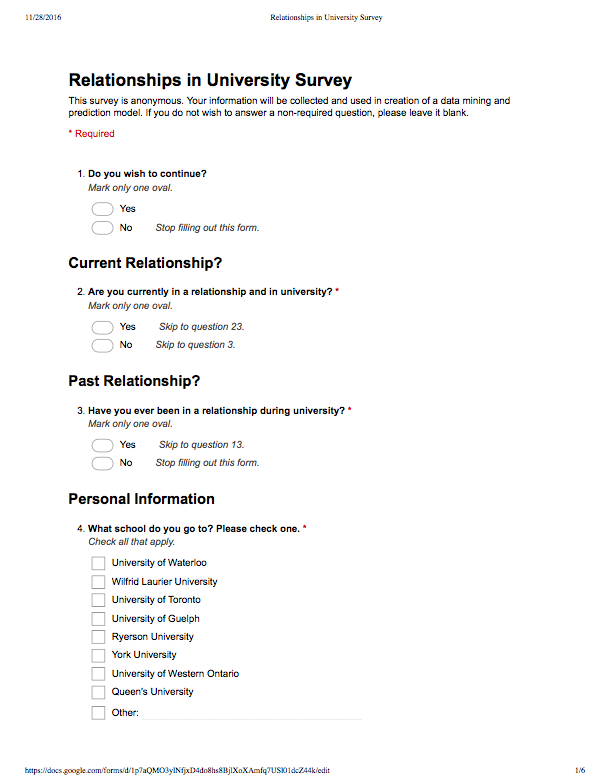
**Related Work**

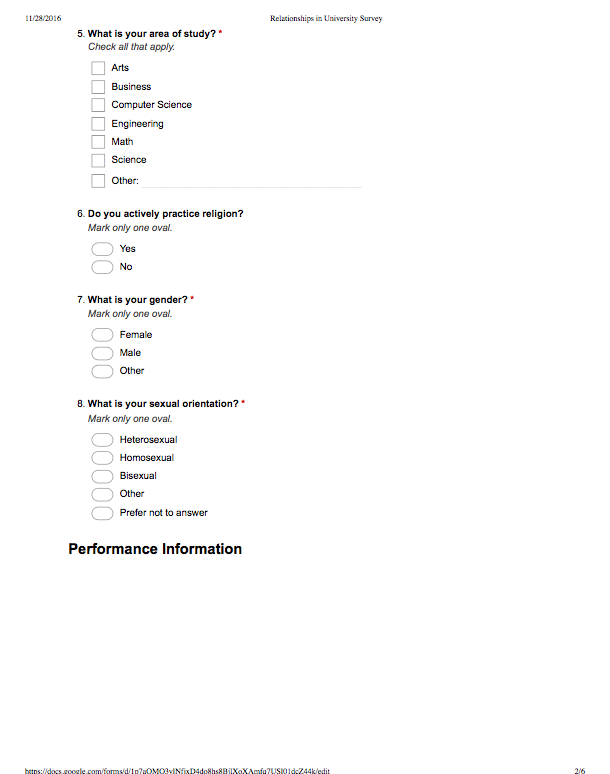
<http://search.proquest.com/openview/768ec6bb9986f2edd98c3a0966f5d1a2/1?pq-origsite=gscholar>

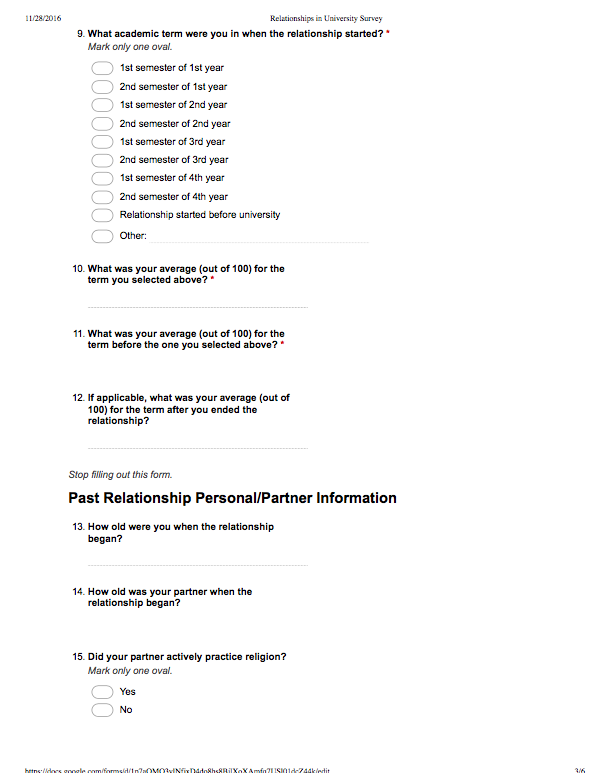
<http://www.sciencedirect.com/science/article/pii/S0049089X07000373>

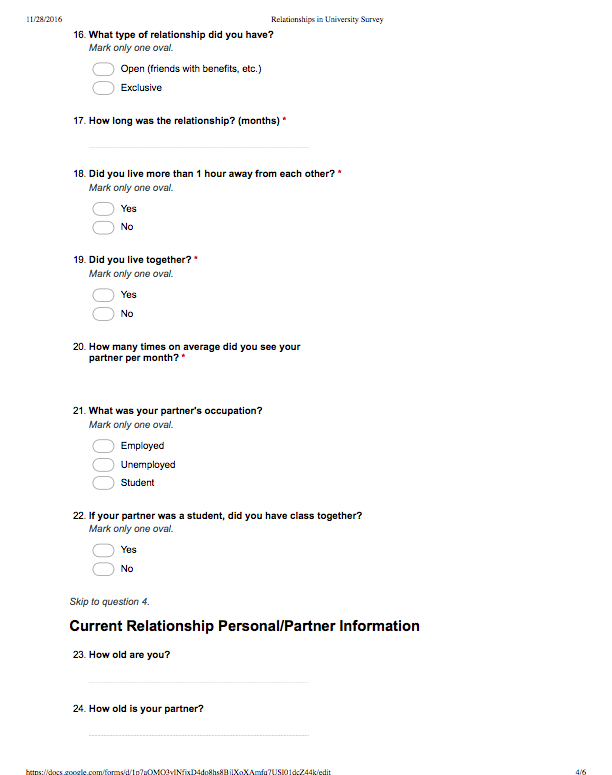
<http://www.ingentaconnect.com/content/prin/csj/2012/00000046/00000003/art00015>

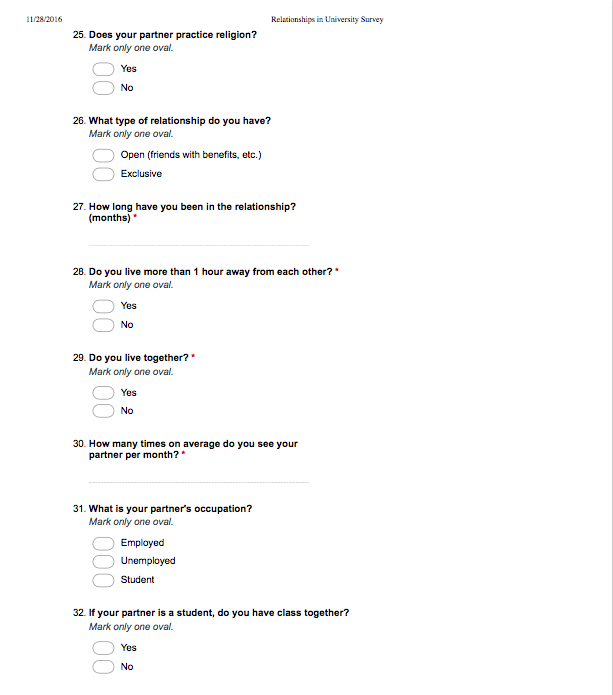
**Appendix A: Survey**











**Appendix B: Code**

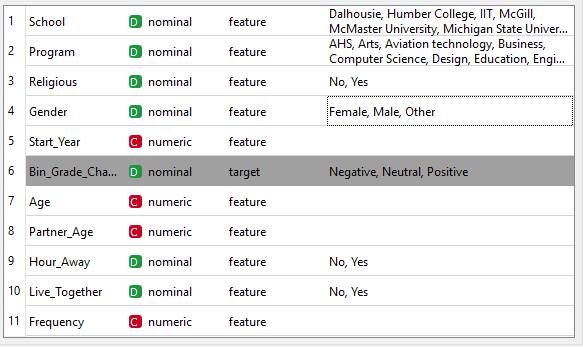
**B-1 Naive bayes**

#adapted from tutorial example code

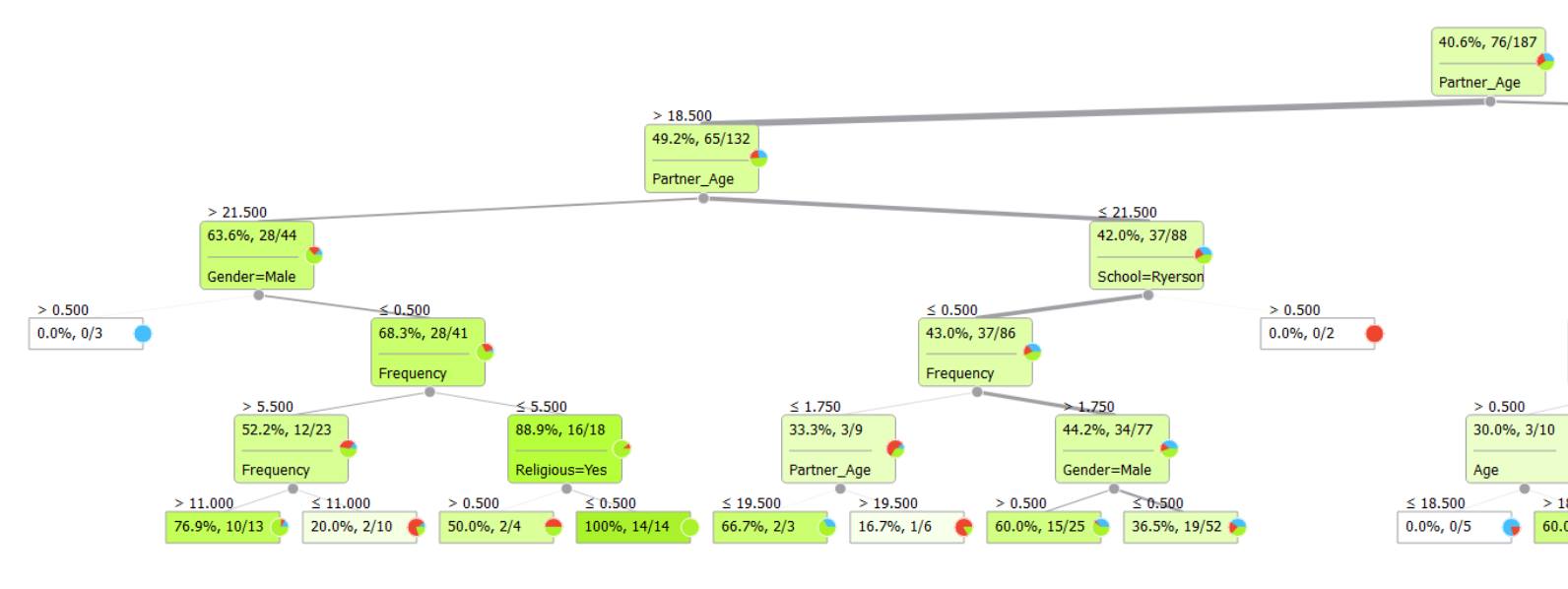
import pandas as pd  
from sklearn import metrics  
from sklearn.naive\_bayes import GaussianNB  
import numpy as np  
from sklearn.cross\_validation import KFold, cross\_val\_score  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn import preprocessing  
  
dataset = pd.read\_csv('446-dataforNB.csv')  
  
Age = dataset.Age.reshape((len(dataset.Age),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Age)  
Agebin = enc.transform(Age).toarray()  
  
School = dataset.SchoolNum.reshape((len(dataset.SchoolNum),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(School)  
Schoolbin = enc.transform(School).toarray()  
  
Program = dataset.ProgramNum.reshape((len(dataset.ProgramNum),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Program)  
Programbin = enc.transform(Program).toarray()  
  
Religion = dataset.IsReligious.reshape((len(dataset.IsReligious),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Religion)  
Relbin = enc.transform(Religion).toarray()  
  
Gender = dataset.IsFemale.reshape((len(dataset.IsFemale),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Gender)  
Genderbin = enc.transform(Gender).toarray()  
  
Distance = dataset.HourBin.reshape((len(dataset.HourBin),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Distance)  
Distbin = enc.transform(Distance).toarray()  
  
P\_Age = dataset.Partner\_Age.reshape((len(dataset.Partner\_Age),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(P\_Age)  
Partnerbin = enc.transform(P\_Age).toarray()  
  
Frequency = dataset.Frequency.reshape((len(dataset.Frequency),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Frequency)  
FreqBin = enc.transform(Frequency).toarray()  
  
LiveTog = dataset.TogetherBin.reshape((len(dataset.TogetherBin),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(LiveTog)  
LiveTogBin = enc.transform(LiveTog).toarray()  
  
Startyear = dataset.Start\_Year.reshape((len(dataset.Start\_Year),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Startyear)  
Startbin = enc.transform(Startyear).toarray()  
  
# prepare datasets to be fed into the naive bayes model  
#predict grade change bin given school and program  
CV = dataset.Grade\_Bin.reshape((len(dataset.Grade\_Bin), 1))  
  
data = np.concatenate((FreqBin,Startbin), axis=1)  
  
#data = np.concatenate((FreqBin,LiveTogBin,Distbin,Genderbin, Programbin,Schoolbin,Agebin,Startyear,Partnerbin, Relbin), axis=1)  
NB = MultinomialNB()  
  
NB.fit(data,CV)  
  
#predict the class for each data point  
predicted = NB.predict(data)  
print("Predictions:\n",np.array([predicted]).T)  
  
# predict the probability/likelihood of the prediction  
prob\_of\_pred = NB.predict\_proba(data)  
print("Probability of each class for the prediction: \n",prob\_of\_pred)  
print("Accuracy of the model: ",NB.score(data,CV))  
  
print("The confusion matrix:\n", metrics.confusion\_matrix(CV, predicted, ['Greater than 4%','No change','Less than -4%','0 to 4%','-4% to 0']))  
  
model = MultinomialNB()  
kf = KFold(len(CV), n\_folds=5)  
scores = cross\_val\_score(model, data, CV, cv=kf)  
print("Accuracy of every fold in 5 fold cross validation: ", abs(scores))  
print("Mean of the 5 fold cross-validation: %0.2f" % abs(scores.mean()))

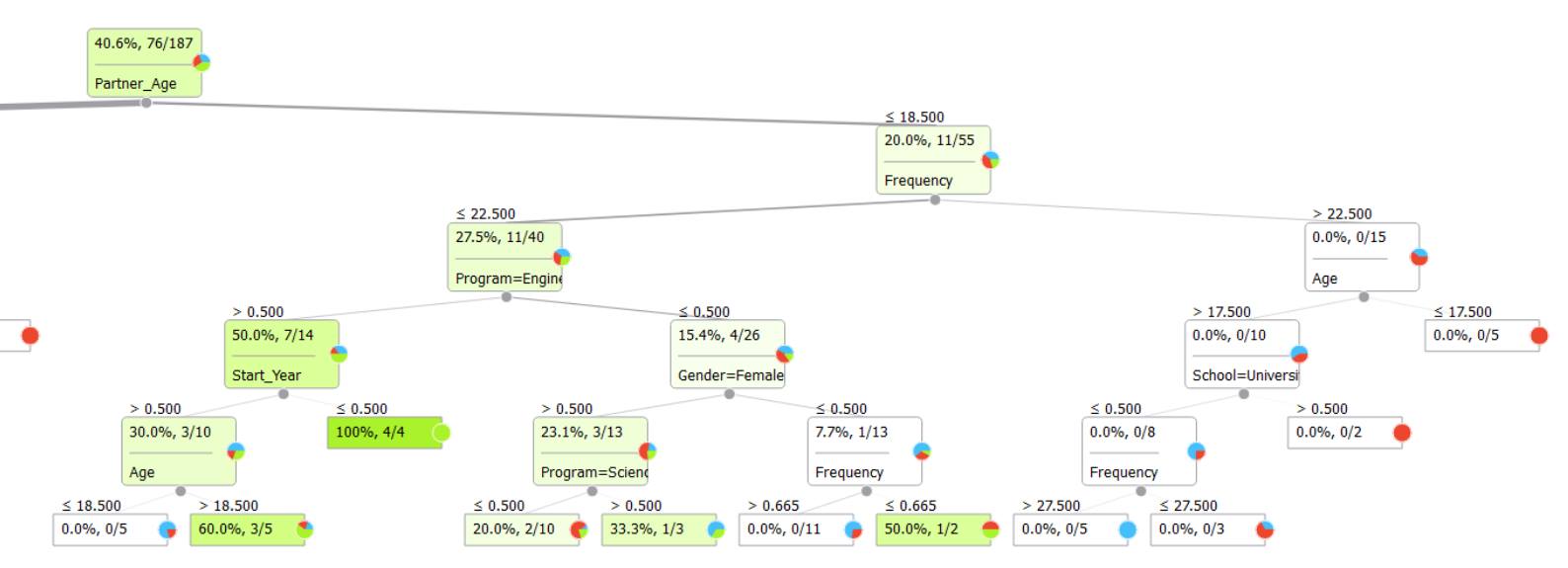
**B-2 Decision trees**

**Input Data A**

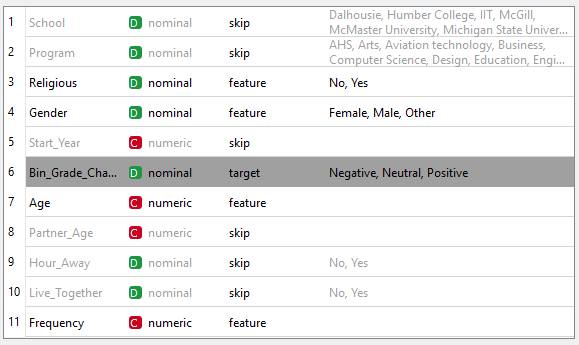


**Data Tree A**

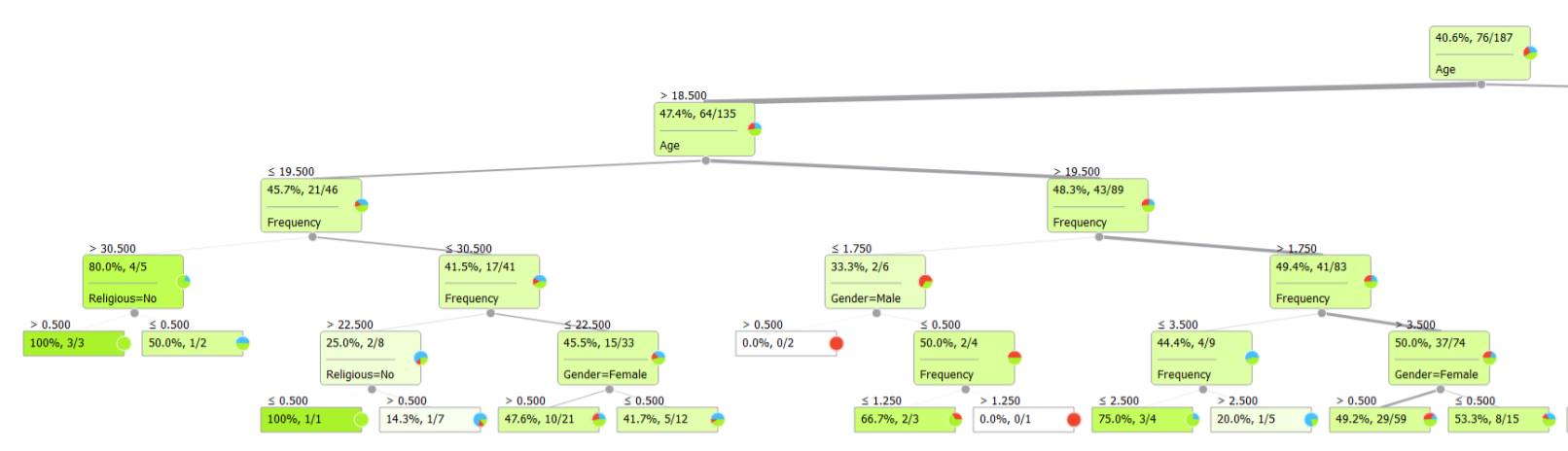


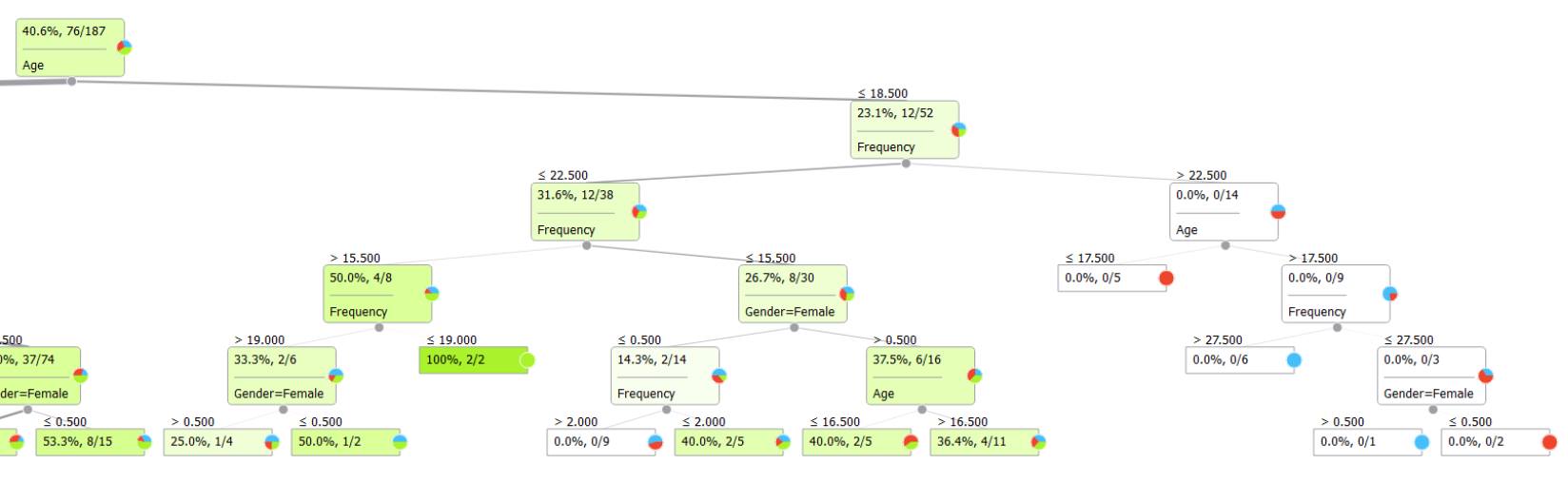


**Input Data B**

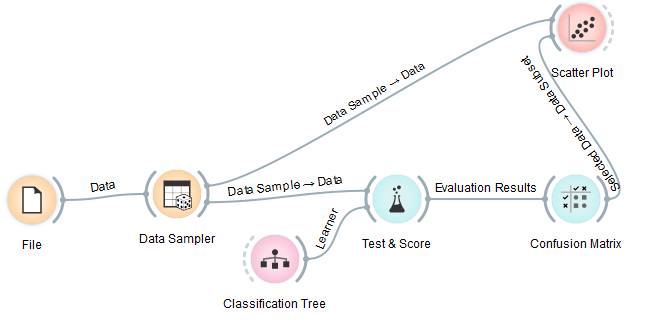


**Data Tree B**





**Orange “Code”**



**B-3 K-Nearest Neighbor**

#modified from Tutorial Example Code

import pandas as pd  
import numpy as np  
from sklearn import metrics  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.cross\_validation import KFold, cross\_val\_score  
  
dataset = pd.read\_csv('knn.csv')  
print(dataset)  
  
# prepare datasets to be fed in the regression model  
#predict grade increase given start year and frequency  
CV = dataset.grade\_increase.reshape((len(dataset.grade\_increase), 1))  
data = (dataset.ix[:,'frequency':'start\_year'].values).reshape((len(dataset.grade\_increase), 2))  
  
# Create a KNN object  
KNN = KNeighborsClassifier(n\_neighbors=3)  
  
# Train the model using the training sets  
KNN.fit(data, CV)  
  
#predict the class for each data point  
predicted = KNN.predict(data)  
print("Predictions: \n", np.array([predicted]).T)  
  
# predict the probability/likelihood of the prediction  
print("Probability of prediction: \n",KNN.predict\_proba(data))  
  
print("Neighbors and their Distance: \n",KNN.kneighbors(data, return\_distance=True))  
  
print("Accuracy score for the model: \n", KNN.score(data,CV))  
  
print(metrics.confusion\_matrix(CV, predicted, labels=["Yes","No"]))  
  
# Calculating 5 fold cross validation results  
model = KNeighborsClassifier()  
kf = KFold(len(CV), n\_folds=5)  
scores = cross\_val\_score(model, data, CV, cv=kf)  
print("Accuracy of every fold in 5 fold cross validation: ", abs(scores))  
print("Mean of the 5 fold cross-validation: %0.2f" % abs(scores.mean()))

**B-4 Linear Regression**

#adapted from code used in Tutorial

import pandas as pd

import matplotlib.pyplot as plt

import math

import pylab as P

import numpy as np

from sklearn import linear\_model

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error

from sklearn import preprocessing

dataset = pd.read\_csv('linear\_regression.csv')

print(dataset)

#plot a scatter plot

plt.figure(1)

plt.scatter(dataset.Age, dataset.Grade\_Change, color='blue')

plt.title("Grade change as a function of age")

plt.xlabel("Age")

plt.ylabel("Grade Change")

plt.show()

# fit a line to this data

# reshape data and CV to be a matrix again

data = dataset.Age.reshape((len(dataset.Age), 1))

CV = dataset.Grade\_Change.reshape((len(dataset.Grade\_Change), 1))

# Create linear regression object

regr = linear\_model.LinearRegression()

# Train the model using the training sets

regr.fit(data, CV)

# get the predictions on the training data

predicted\_results = regr.predict(data)

print("Results with outlier:")

# The coefficients (m, b) of y = mx + b

print('Coefficients (m): \n', regr.coef\_)

print('Intercept (b): \n', regr.intercept\_)

# The mean square error MSE or the mean residual sum of square of errors should be less

MSE = mean\_squared\_error(CV,predicted\_results)

RMSE = math.sqrt(MSE)

# Explained variance score: 1 is perfect prediction

R2 = r2\_score(CV,predicted\_results)

print("Mean residual sum of squares =", MSE)

print("RMSE =", RMSE)

print("R2 =", R2)

print("Mean residual sum of squares = %.2f"

% np.mean((regr.predict(data) - CV) \*\* 2))

print('R2 = %.2f' % regr.score(data,CV))

plt.plot(data, predicted\_results, color='green', linewidth=3)

plt.scatter(data, CV, color='black')

plt.xlabel("age")

plt.ylabel("grade\_change")

plt.show()

# to see how the residual errors behave

residual\_error = CV - predicted\_results

print("Mean of residuals =", np.mean(residual\_error))

print("Standard deviation of residuals =", np.std(residual\_error))

plt.figure(3)

plt.plot((-40,140),(0,0), 'r--')

plt.scatter(data,residual\_error,label='residual error')

plt.title("Residual plot")

plt.xlabel("age")

plt.ylabel("grade\_change")

plt.show()

plt.figure(4)

plt.hist(residual\_error)

plt.title("Distribution of residuals")

plt.xlabel("residual error")

plt.show()

plt.figure(5)

n, bins, patches = plt.hist(residual\_error, 10, normed=1, alpha = 0.5)

y\_pdf = P.normpdf(bins, np.mean(residual\_error), np.std(residual\_error))

l = P.plot(bins, y\_pdf, 'k--', linewidth=1.5)

plt.show()

print("If a person is 18 years old, what do you predict about their grade change: %.2f" % regr.predict(18))

print("If a person is 20 years old, what do you predict about their grade change: %.2f" % regr.predict(20))

print("If a person is 22 years old, what do you predict about their grade change: %.2f" % regr.predict(22))

**B-5 Multi-regression**

#adapted from tutorial example code

import seaborn as sns  
import pandas as pd  
import matplotlib.pyplot as plt  
import math  
import pylab as P  
import numpy as np  
  
from sklearn import linear\_model  
from sklearn.metrics import r2\_score  
from sklearn.metrics import mean\_squared\_error  
from sklearn import preprocessing  
  
dataset = pd.read\_csv('446-dataforNB.csv')  
print(dataset)  
  
Age = dataset.Age.reshape((len(dataset.Age),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Age)  
Agebin = enc.transform(Age).toarray()  
  
School = dataset.SchoolNum.reshape((len(dataset.SchoolNum),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(School)  
Schoolbin = enc.transform(School).toarray()  
  
Program = dataset.ProgramNum.reshape((len(dataset.ProgramNum),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Program)  
Programbin = enc.transform(Program).toarray()  
  
Religion = dataset.IsReligious.reshape((len(dataset.IsReligious),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Religion)  
Relbin = enc.transform(Religion).toarray()  
  
Gender = dataset.IsFemale.reshape((len(dataset.IsFemale),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Gender)  
Genderbin = enc.transform(Gender).toarray()  
  
Distance = dataset.HourBin.reshape((len(dataset.HourBin),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Distance)  
Distbin = enc.transform(Distance).toarray()  
  
P\_Age = dataset.Partner\_Age.reshape((len(dataset.Partner\_Age),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(P\_Age)  
Partnerbin = enc.transform(P\_Age).toarray()  
  
Frequency = dataset.Frequency.reshape((len(dataset.Frequency),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Frequency)  
FreqBin = enc.transform(Frequency).toarray()  
  
LiveTog = dataset.TogetherBin.reshape((len(dataset.TogetherBin),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(LiveTog)  
LiveTogBin = enc.transform(LiveTog).toarray()  
  
Startyear = dataset.Start\_Year.reshape((len(dataset.Start\_Year),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Startyear)  
Startbin = enc.transform(Startyear).toarray()  
  
# prepare datasets to be fed in the regression model  
CV = dataset.Grade\_Change.reshape((len(dataset.Grade\_Change), 1))  
  
# This is what you edit for variables to include

comparevar = Startbin  
  
othervars = [FreqBin,LiveTogBin,Distbin,Genderbin, Programbin,Schoolbin,Agebin,Startyear,Partnerbin, Relbin]  
x = 0  
y = 0  
percentage =0  
for exvar in othervars:  
 data = np.concatenate((Distbin,Partnerbin,Programbin,FreqBin,Schoolbin,Agebin,Startbin), axis=1)  
 regr = linear\_model.LinearRegression()  
 regr.fit(data, CV)  
 predicted\_results = regr.predict(data)  
 print(x,': ',regr.score(data,CV))  
 if regr.score(data,CV) > percentage:  
 y = x  
 percentage = regr.score(data,CV)  
 x = x+1  
  
print()  
print(y, ' - ', percentage)  
  
#print("The processed dataset: ", np.concatenate((data, CV), axis=1))  
  
  
print("Results:")  
# # The coefficients (m, b) of y = mx + b  
print('Coefficients (m1, m2, m3): \n', regr.coef\_)  
print('Intercept (b): \n', regr.intercept\_)  
#  
print("Mean residual sum of squares = %.2f"  
% np.mean((regr.predict(data) - CV) \*\* 2))  
print('R2 = %.3f' % percentage)  
  
# to see how the residual errors behave  
residual\_error = CV - predicted\_results  
print("Mean of residuals =", np.mean(residual\_error))  
print("Standard deviation of residuals =", np.std(residual\_error))

**B-6 Logistic Regression**

#adapted from tutorial example code

import pandas as pd  
import numpy as np  
from sklearn import metrics  
from sklearn.linear\_model import LogisticRegression  
from sklearn.cross\_validation import KFold, cross\_val\_score  
from sklearn import preprocessing  
  
dataset = pd.read\_csv('446-data-logisticregression.csv')  
print(dataset)  
  
Age = dataset.Age.reshape((len(dataset.Age),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Age)  
Agebin = enc.transform(Age).toarray()  
  
School = dataset.SchoolNum.reshape((len(dataset.SchoolNum),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(School)  
Schoolbin = enc.transform(School).toarray()  
  
Program = dataset.ProgramNum.reshape((len(dataset.ProgramNum),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Program)  
Programbin = enc.transform(Program).toarray()  
  
Religion = dataset.IsReligious.reshape((len(dataset.IsReligious),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Religion)  
Relbin = enc.transform(Religion).toarray()  
  
Gender = dataset.IsFemale.reshape((len(dataset.IsFemale),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Gender)  
Genderbin = enc.transform(Gender).toarray()  
  
Distance = dataset.HourBin.reshape((len(dataset.HourBin),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Distance)  
Distbin = enc.transform(Distance).toarray()  
  
P\_Age = dataset.Partner\_Age.reshape((len(dataset.Partner\_Age),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(P\_Age)  
Partnerbin = enc.transform(P\_Age).toarray()  
  
Frequency = dataset.Frequency.reshape((len(dataset.Frequency),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Frequency)  
FreqBin = enc.transform(Frequency).toarray()  
  
LiveTog = dataset.TogetherBin.reshape((len(dataset.TogetherBin),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(LiveTog)  
LiveTogBin = enc.transform(LiveTog).toarray()  
  
Startyear = dataset.Start\_Year.reshape((len(dataset.Start\_Year),1))  
enc = preprocessing.OneHotEncoder()  
enc.fit(Startyear)  
Startbin = enc.transform(Startyear).toarray()  
  
data = np.concatenate((Frequency,Startbin,Genderbin), axis=1)  
CV = dataset.GradeDecrease.reshape((len(dataset.GradeDecrease),1))  
LogReg = LogisticRegression()  
model = LogisticRegression()  
LogReg.fit(data, CV)  
predicted = LogReg.predict(data)  
modelscore = LogReg.score(data,CV)  
#print("Accuracy score for the model: \n", LogReg.score(data,CV))  
kf = KFold(len(CV), n\_folds=5)  
scores = cross\_val\_score(model, data, CV, cv=kf)  
kscoreout = abs(scores.mean())  
x = 0  
diffone = abs(kscoreout-modelscore)  
print("Accuracy score for the model: \n", LogReg.score(data, CV))  
print("Accuracy of every fold in 5 fold cross validation: ", abs(scores))  
print("Mean of the 5 fold cross-validation: %0.2f" % abs(scores.mean()))  
  
# the model  
print('Coefficients (m): \n', LogReg.coef\_)  
print('Intercept (b): \n', LogReg.intercept\_)  
  
#predict the class for each data point  
predicted = LogReg.predict(data)  
print("Predictions: \n", np.array([predicted]).T)  
  
# predict the probability/likelihood of the prediction  
print("Probability of prediction: \n",LogReg.predict\_proba(data))  
  
print()  
print(metrics.confusion\_matrix(CV, predicted, labels=["Yes","No"]))  
print("Female, 1, Year 1: ", LogReg.predict([1,0,1,0,0,0,0,1]),LogReg.predict\_proba([1,0,1,0,0,0,0,1]))  
print("Male, 1, Year 1: ", LogReg.predict([1,0,1,0,0,0,1,0]),LogReg.predict\_proba([1,0,1,0,0,0,1,0]))  
print("Female, 1, Year 1: ", LogReg.predict([5,0,1,0,0,0,0,1]),LogReg.predict\_proba([5,0,1,0,0,0,0,1]))  
print("Male, 5, Year 1: ", LogReg.predict([5,0,1,0,0,0,1,0]),LogReg.predict\_proba([5,0,1,0,0,0,1,0]))  
print("Female, 1, Year 1: ", LogReg.predict([10,0,1,0,0,0,0,1]),LogReg.predict\_proba([10,0,1,0,0,0,0,1]))  
print("Male, 10, Year 1: ", LogReg.predict([10,0,1,0,0,0,1,0]),LogReg.predict\_proba([10,0,1,0,0,0,1,0]))  
print("Female, 1, Year 1: ", LogReg.predict([15,0,1,0,0,0,0,1]),LogReg.predict\_proba([15,0,1,0,0,0,0,1]))  
print("Male, 15, Year 1: ", LogReg.predict([15,0,1,0,0,0,1,0]),LogReg.predict\_proba([15,0,1,0,0,0,1,0]))  
print("Female, 1, Year 1: ", LogReg.predict([20,0,1,0,0,0,0,1]),LogReg.predict\_proba([20,0,1,0,0,0,0,1]))  
print("Male, 20, Year 1: ", LogReg.predict([20,0,1,0,0,0,1,0]),LogReg.predict\_proba([20,0,1,0,0,0,1,0]))  
print("Female, 1, Year 1: ", LogReg.predict([25,0,1,0,0,0,0,1]),LogReg.predict\_proba([25,0,1,0,0,0,0,1]))  
print("Male, 25, Year 1: ", LogReg.predict([25,0,1,0,0,0,1,0]),LogReg.predict\_proba([25,0,1,0,0,0,1,0]))  
print("Female, 1, Year 1: ", LogReg.predict([30,0,1,0,0,0,0,1]),LogReg.predict\_proba([30,0,1,0,0,0,0,1]))  
print("Male, 30, Year 1: ", LogReg.predict([30,0,1,0,0,0,1,0]),LogReg.predict\_proba([30,0,1,0,0,0,1,0]))  
print("Female, 30, Year 1: ", LogReg.predict([30,0,1,0,0,0,0,1]),LogReg.predict\_proba([30,0,1,0,0,0,0,1]))  
  
print(LogReg.coef\_)

**B-7 Association Rule Mining**

#adapted from tutorial example code

def subsets(arr):  
 return chain(\*[combinations(arr, i + 1) for i, a in enumerate(arr)])  
  
def returnItemsWithMinSupport(itemSet, transactionList, minSupport, freqSet):  
 \_itemSet = set()  
 localSet = defaultdict(int)  
  
 for item in itemSet:  
 for transaction in transactionList:  
 if item.issubset(transaction):  
 freqSet[item] += 1  
 localSet[item] += 1  
  
 for item, count in localSet.items():  
 support = float(count)/len(transactionList)  
  
 if support >= minSupport:  
 \_itemSet.add(item)  
  
 return \_itemSet  
  
def joinSet(itemSet, length):  
 return set([i.union(j) for i in itemSet for j in itemSet if len(i.union(j)) == length])  
  
def getItemSetTransactionList(data\_iterator):  
 transactionList = list()  
 itemSet = set()  
 for record in data\_iterator:  
 transaction = frozenset(record)  
 transactionList.append(transaction)  
 for item in transaction:  
 itemSet.add(frozenset([item])) # Generate 1-itemSets  
 return itemSet, transactionList  
  
def runApriori(data\_iter, minSupport, minConfidence):  
 itemSet, transactionList = getItemSetTransactionList(data\_iter)  
  
 freqSet = defaultdict(int)  
 largeSet = dict()  
 # Global dictionary which stores (key=n-itemSets,value=support)  
 # which satisfy minSupport  
  
 assocRules = dict()  
 # Dictionary which stores Association Rules  
  
 oneCSet = returnItemsWithMinSupport(itemSet, transactionList,minSupport,freqSet)  
 currentLSet = oneCSet  
 k = 2  
 while(currentLSet != set([])):  
 largeSet[k-1] = currentLSet  
 currentLSet = joinSet(currentLSet, k)  
 currentCSet = returnItemsWithMinSupport(currentLSet,  
 transactionList,  
 minSupport,  
 freqSet)  
 currentLSet = currentCSet  
 k = k + 1  
  
 def getSupport(item):  
 """local function which Returns the support of an item"""  
 return float(freqSet[item])/len(transactionList)  
  
 toRetItems = []  
 for key, value in largeSet.items():  
 toRetItems.extend([(tuple(item), getSupport(item))  
 for item in value])  
  
 toRetRules = []  
 for key, value in list(largeSet.items()):  
 for item in value:  
 \_subsets = map(frozenset, [x for x in subsets(item)])  
 for element in \_subsets:  
 remain = item.difference(element)  
 if len(remain) > 0:  
 confidence = getSupport(item)/getSupport(element)  
 if confidence >= minConfidence:  
 toRetRules.append(((tuple(element), tuple(remain)),  
 confidence))  
 return toRetItems, toRetRules  
  
  
def printResults(items, rules):  
 """prints the generated itemsets sorted by support and the confidence rules sorted by confidence"""  
 for item, support in sorted(items, key=lambda support: support[1]):  
 print ("item: %s , %.3f" % (str(item), support))  
 print("------------------------ RULES:")  
 for rule, confidence in sorted(rules, key=lambda rule\_confidence: rule\_confidence[1]):  
 pre, post = rule  
 #if "%" in str(post):  
 print ("Rule: %s ==> %s , %.3f" % (str(pre), str(post), confidence))  
  
def dataFromFile(fname):  
 """Function which reads from the file and yields a generator"""  
 file\_iter = open(fname, 'rU')  
 for line in file\_iter:  
 line = line.strip().rstrip(',') # Remove trailing comma  
 record = frozenset(line.split(','))  
 yield record  
  
inFile = dataFromFile('446-dataforApriori.csv')  
minSupport = 0.1  
minConfidence = 0.9  
  
items, rules = runApriori(inFile, minSupport, minConfidence)  
  
printResults(items, rules)

**B-8 Clustering**

*Clustering using all explanatory variables*

|  |  |
| --- | --- |
| K | sum of distances of samples to their closest cluster center |
| 1 | 23944 |
| 2 | 18820 |
| 3 | 14447 |
| 4 | 11668 |
| 5 | 9216.7 |
| 6 | 8022.2 |
| 7 | 7219.3 |
| 8 | 6594.1 |
| 9 | 6128.3 |
| 10 | 5788.7 |
| 11 | 5429 |
| 12 | 5375 |
| 13 | 5020.3 |
| 14 | 4722 |
| 15 | 4709.3 |
| 16 | 4307.7 |
| 17 | 4231.1 |
| 18 | 3943.7 |
| 19 | 3883.7 |
| 20 | 3686.5 |
| 21 | 3594.9 |
| 22 | 3450.2 |
| 23 | 3336.1 |
| 24 | 3141.7 |
| 25 | 3104.4 |
| 26 | 3000 |
| 27 | 2841.2 |
| 28 | 2832.9 |
| 29 | 2730.9 |
| 30 | 11668 |

#Modified from the Example Tutorial Code

from sklearn.cluster import KMeans  
import pandas as pd  
import matplotlib.pyplot as plt  
import matplotlib  
from mpl\_toolkits.mplot3d import Axes3D  
  
dataset = pd.read\_csv('k\_means\_all.csv')  
print(dataset)  
  
KM = KMeans(n\_clusters=10, init='k-means++', random\_state=170)  
KM = KM.fit(dataset)  
  
print("The cluster centroids are: \n", KM.cluster\_centers\_)  
print("Cluster", KM.labels\_)  
print("Sum of distances of samples to their closest cluster center: ", KM.inertia\_)