Exploratory Data Analysis

Data exploration analysis is a critical component to the success of any major research project. The goal is to uncover and summarize the main characteristics of the dataset. This provides the researcher a more holistic understanding of the variables and relationships within the dataset. Data exploration analysis is often the first step in a major research project and usually includes data wrangling, data visualization, and feature transformation.

The first major data exploration step that I applied to my dataset was principle component analysis (PCA). PCA is a dimensionality reduction technique that is used to identify a small number of uncorrelated variables called “principles components” from a large dataset. The goal of PCA is to represent the maximum amount of variance in a dataset with the fewest number of principle components. As noted by Bro & Smilde [1], Principal component analysis is one of the most important and powerful method used in statistics. When working with Bayesian networks, PCA is an extremely valuable tool because it can be used to reduce the number of nodes in the directed acyclic graph. Each feature in the dataset corresponds to a node in the Bayesian network, therefore reducing the number of features significantly lowers the computational time of inferencing. Through my experimentation, I decided to limit the number of nodes in my network to 15, which comprise of the first 13 principle components, the class, and the dollar amount.

The next major data preprocessing step that I applied to my dataset is feature normalization. Feature normalization or feature scaling is a data mining technique used to standardize the range of independent variables in the dataset. “Feature normalization is an important aspect of the classification process, since it effectively changes the characteristics of

the underlying probability distributions", Davatzikos et al [2]. I normalized each of my principle components into a range of [0,1] using the following formula:

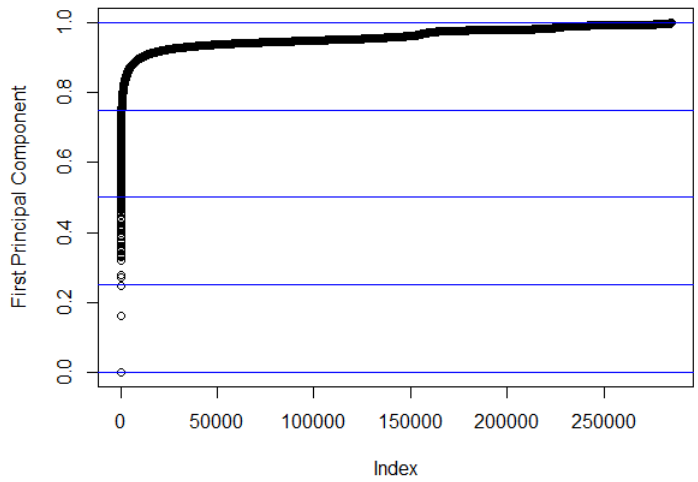
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

In addition to the standard benefits of feature normalization, it also provides a much more streamlined approach to data discretization. The next step in the data preprocessing phase that I applied to my dataset was variable discretization. Feature discretization is the process of converting continuous variables into discrete variables. "Many real-world classification tasks exist that involve continuous features where such algorithms could not be applied unless the continuous features are first discretized" Dougherty et al [3]. This is exactly the case when working with Bayesian networks. Bayesian networks in R are only able to function if all of the variables in the dataset are of type factor. In order to achieve this, I experimented with multiple different discretization methods. Initially, I attempted to bin my variables based on the median of each of my principle components. I assigned a value of 1 to any value above the median and a value of 0 to anything below the median. After applying this strategy and experimenting with Bayesian network inferencing, it was determined that it significantly reduces the model's accuracy and precision. Next, I attempted to discretize my principle components by binning them into 1s and 0s based on the clusters of the positive cases in each column. Again, this discretization strategy resulted in a significant loss of inference accuracy and precision. Finally, I decided to bin my columns into 4 separate quadrants, ([0:0.25], [0.25:0.5], [0.5:0.75], [0.75:1]). It was determined that is discretization strategy proved the highest accuracy and precision, while still maintaining a reasonable computing time.

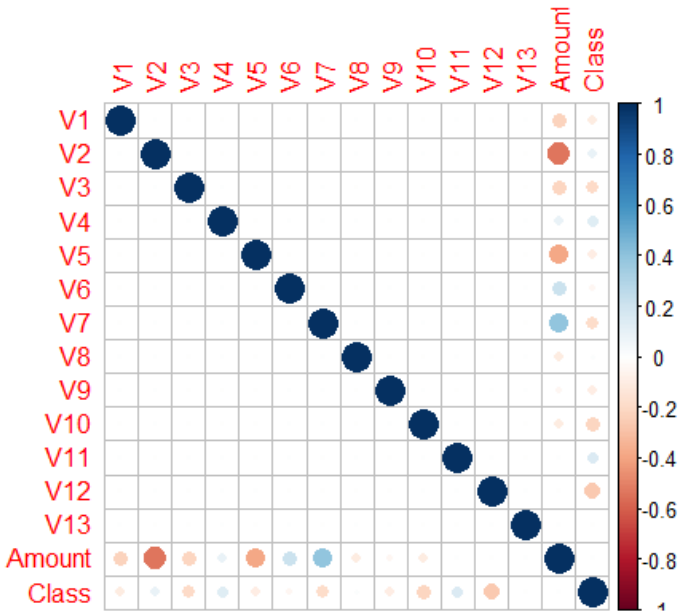
Ideally, I would have liked to further granularize my principle components into a larger number of bins, however a computational trade off exists. As I increase the number of bins model accuracy and precision also increase, however this is at the cost of computational time. In a Bayesian network, each additional bin results in an extra conditional probability calculation for each record in the model. Therefore, the goal is to find the optimal number of bins, that would result in high model accuracy and precision, while minimizing the computational time. In addition to this trade off, when discretizing continuous variables there always exists a discretization error. Discretization error is the error that results from transforming a continuous variable into a discrete variable. This error can be represented by the following equation:

$$Error = \frac{Continuous - Discrete}{Discrete}$$

Given the fact that I am using 4 bins of width 0.25 the average discretization error that exists in my dataset is 50%. The following figure graphically represents the discretization boundaries that I implemented on each of the principal components in my dataset (see plot below).



The next major data exploration step that I applied to my dataset is attempting to understand the correlations that exists within my dataset. Acquiring a strong understanding of the correlations that exists in the dataset is an essential component to developing an accurate and precise machine learning classification model. In order to acquire this understating, I constructed a multiple variable correlation matrix. This matrix was used to analysis the relationships between variables and the predicting class (see plot below). As you can see there exists correlations between a majority of the variables and the class/amount variable. There does not exist any relationships between principle components, as expected.



Finally, the last data preprocessing step that I applied was splitting my data into testing and training sets. After experimenting with various data splitting ratios, I decided to split my data ~88% training (250000 observations) and ~12% testing (34807 observations). When working with Bayesian networks in R, the training set must contain all factors levels in each of

the inputted columns. Therefore, I am constrained to either include a large training set or reduce the number of bins in the dataset. Reducing the number of bins would result in a loss of accuracy and precision. In addition, due to the highly-unbalanced class distribution in my dataset, it is vital that my training set contains an adequate number of positive cases. Thus, proceeding with a large training set is a more appropriate option.

Literature Review

The number of credit card transactions are growing, taking an ever-larger share of the worlds payment system. As noted by Chan et al [4], the increase in credit card transactions are leading to higher rates of stolen account numbers and subsequent losses by banks. Improved credit card fraud detection techniques are required to maintain the viability of the worlds payment system. Banks have used early fraud warning systems for many years, however these systems are not sufficient in preventing all types of fraud. Advances in data mining and machine learning have opened the door for massive large scale improvements in credit card fraud detection. In addition, computing technologies have significantly improved allowing individuals to "analyze massive amounts of transactions data" Chan et al [4]. As noted by Chan et al [4], one of the major challenges when dealing with credit card fraud data is the class imbalance. Due to the nature of fraud data, most of the classes are negative and only a very small portion of the classes are positive.

"A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest" Heckerman [5]. When used in conjunction with statistical techniques, the graphical model has several advantages for data analysis and classification. As noted by Friedman et al [6] learning Bayesian networks is often cast as an optimization problem, where

the computational task is to find a structure that maximizes a statistically motivated score. Maxwell et al [7] and Gamez et al [8] mentioned that learning the structure of a Bayesian network from large dataset is known to be an NP-Hard problem. In many different domains, the application of a heuristic search has proven to be an advantageous strategy in learning Bayesian network structures. The heuristic approach is computationally efficient and, even though it does not guarantee an optimal result, many previous studies have shown that it obtains very good solutions. For this project, I implemented a greedy search algorithm known as the hill climbing algorithm. “Hill climbing algorithms are particularly popular because of their good trade-off between computational demands and the quality of the models learned” Gamez et al [8]. My Bayesian network will also follow a variation of the Tree Augmented Naïve Bayes (TAN) Structure, where the root node will be my predicting class. “Tree Augmented Naive Bayes (TAN), which outperforms naive Bayes, yet at the same time maintains the computational simplicity (no search involved) and robustness that characterize naive Bayes” Friedman et al [9].

Bayesian networks are considered to be complete models and therefore can be used to answer probabilistic queries. This process involves computing the posterior distribution of variables given evidence, and is known as probabilistic inferencing. Bayesian networks have and are currently being used in many different domains for probabilistic inferencing. As noted by Kirkos et al [10], Bayesian networks are extremely effective models for predicting fraudulent financial statements and identifying associated factors. In terms of performance, Bayesian Belief networks out perform all other inferencing algorithms when working with financial data. In past studies, Maes et al [11], Bayesian networks have been used to predict fraudulent credit card transactions, and have proved to be effective in dealing with the large class imbalance.

However, as noted by Maes et al [11], the fraud detection process when using Bayesian networks is considered to be slow in comparison with other machine learning algorithms.

Researchers have been and continue to seek out incremental improvements in algorithm efficiency. As the speed at which sequential computing begins to plateau, the logical solution is to scale computing power horizontally. Hence, we see the introduction to parallel computing. “In order to solve a problem efficiently on a parallel machine, it is usually necessary to design an algorithm that specifies multiple operations on each step, i.e., a parallel algorithm” Blelloch & Maggs [12]. Bayesian networks are an example of an algorithm that has the capabilities to be split on multiple processing cores. As noted by Madsen & Jensen [13], one way of improving the time efficient of Bayesian networks is to take advantage of the inference algorithm’s available parallelism. The inference algorithm relies on a Monte Carlo simulation, which can easily be parallelized, as seen in Maigne et al [14].

“Many supervised/unsupervised machine learning algorithms require a discrete feature space” Dougherty et al [3]. Bayesian networks are an example of an machine learning algorithm that requires all features to be factors, i.e. discrete variables. As noted by Dougherty et al [3], Equal Interval Width discretization merely divides the range of observed values for a variable into k equally sized bins, where k is a user-supplied parameter. As mentioned in Kerber [15], Equal Interval Width discretization does not utilize instance labels in setting partition boundaries, therefore it is likely that classification information will be lost by binning values that are strongly associated with different classes into the same bin. This discretization error can be minimized by increasing the total number of bins.

References

[1] R. Bro and A. Smilde, "Principal component analysis", Analytical Methods, vol. 6, no. 9, p. 2812, 2014.

[2] C. Davatzikos, K. Ruparel, Y. Fan, D. Shen, M. Acharyya, J. Loughead, R. Gur and D. Langleben, "Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection", NeuroImage, vol. 28, no. 3, pp. 663-668, 2005.

[3] J. Dougherty, R. Kohavi and M. Sahami, MACHINE LEARNING: PROCEEDINGS OF THE TWELFTH INTERNATIONAL CONFERENCE: Supervised and Unsupervised Discretization of Continuous Features, 1st ed. Palo Alto: Morgan Kaufmann, 1995, pp. 194-202.

[4] P. Chan, W. Fan and A. Prodromidis, "Distributed data mining in credit card fraud detection", IEEE Intelligent Systems, vol. 14, no. 6, pp. 67-74, 1999.

[5] D. Heckerman, Learning in Graphical Models: A Tutorial on Learning with Bayesian Networks, 1st ed. Cambridge: Springer Netherlands, 1998, pp. 301-354.

[6] N. Friedman, I. Nachman and D. Peér, "Learning bayesian network structure from massive datasets", artificial intelligence, pp. 206-215, 1999.

[7] D. Maxwell, D. Heckerman and C. Meek, "Large-Sample Learning of Bayesian Networks is NP-Hard", Machine Learning Researc, vol. 17, pp. 1287-1330, 2004.

[8] J. Gámez, J. Mateo and J. Puerta, "Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood", Data Mining and Knowledge Discovery, vol. 22, no. 1-2, pp. 106-148, 2010.

[9] N. Friedman, D. Geiger and M. Goldszmidt, "Bayesian Network Classifiers", Machine Learning, vol. 29, no. 23, pp. 131-163, 1997.

[10] E. KIRKOS, C. SPATHIS and Y. MANOLOPOULOS, "Data Mining techniques for the detection of fraudulent financial statements", Expert Systems with Applications, vol. 32, no. 4, pp. 995-1003, 2007.

[11] S. Maes, K. Tuyls, B. Vanschoenwinkel and B. Manderick, "Credit Card Fraud Detection Using Bayesian and Neural Networks", American Association Neurological Surgeons, pp. 261-270, 1993.

[12] G. Blelloch and B. Maggs, Algorithms and theory of computation handbook, 2nd ed. Chapman & Hall/CRC, 2010, pp. 25-25.

[13] A. Madsen, F. Jensen, A. Salmerón, H. Langseth and T. Nielsen, "A parallel algorithm for Bayesian network structure learning from large data sets", Knowledge-Based Systems, vol. 117, pp. 46–55, 2017.

[14] L. MAIGNE, D. HILL, P. CALVAT, V. BRETON, R. REUILLON, D. LAZARO, Y. LEGRE and D. DONNARIEIX, "PARALLELIZATION OF MONTE CARLO SIMULATIONS AND SUBMISSION TO A GRID ENVIRONMENT", Parallel Processing Letters, vol. 14, no. 02, pp. 177-196, 2004.

[15] R. Kerber, "discretization of numeric attributes", Artificial intelligence, pp. 123-128, 1992.