Research on Probability-based Learning Application on Car Insurance Data

Longhao Jing[1, a], Wenjing Zhao[1, b], Karthik Sharma[1, c], Runhua Feng[1, d *]

[1] Computer Science, University of Illinois at Urbana-Champaign (UIUC), 61820, United States of America

[a]email, [b]email, [c]email, [d]email

* Corresponding Author: Runhua Feng

Keywords: Machine learning, Probability-based learning; Insurance risk, R

Abstract. After entering the big data era, there is an increasing demand on data analysis. It is natural for the modern actuary to question tech buzzwords like “machine learning” and “data analytics.” In reality, many machine-learning models have a basis in the very concepts, which actuaries have used to assess risk for a long time. We refer to the machine learning techniques that deal most explicitly with probabilities and risks as probability-based learning, and will focus on applying probability-base learning models on a set of car insurance data to create an artificial intelligence to accelerate the underwriting process for property and casualty insurance pricing.

Introduction

The goal of probability-based learning is given some relevant information and the model is to inform us of a range of probable outcomes. In other words, we would like our model to train on data such that it may accurately predict the value of a target feature in a data set when given the values taken by the set of features, which may influence it. We refer to these features as descriptive features.

Problem Analysis

One of the potential flaws of probability-based learning is that there is no way to analyze the interdependency between the descriptive features. In theory, we usually assume them to be independent from each other. However, in reality it is usually not the case. Therefore, to help our model identify these relationships, we must choose the data set that best represents the range of possible feature-value assignments for which the model must prepare. In this case, we choose the Naïve Bayes’ model and the Bayesian Network model to analyze the car insurance.

Naïve Bayes’ Model. The first model we will cover is the Naïve Bayes’ model, whose “naïveté” lies in its assumption of conditional independence between descriptive as just described. When the model has learned each of the probabilities involved in the formula, it returns the most likely value taken by our target feature by applying the following function

$$\max_{v\, \in \mathrm{values}(t)} \left(\left(\prod_{i=1}^{m} P(d[i] \mid t = v) \right) \times \; P(t = v) \right)$$

Here, our “naive” assumption offers probability-based machine learning models a greater tolerance for data fragmentation.

Data fragmentation is the gradual insufficiency of a given data set as the set of descriptive features in our joint probabilities increases. For example, let’s assume that our model can succesfully compute the probability of five descriptive features taking on a specific set of five values. Given this, there is no guarantee of its success in introducing more variables as there maybe co-corrence between these variables in the data set. The Naïve Bayes’ model readily combats this issue with its assumption of conditional independence.

Though this model may handle insufficient data sets efficiently, it is still not immune to data fragmentation. For example, if there is no instance of a feature taking on a specific value, and we are told to calculate a probability involving that feature taking on that value, its corresponding factor would be equal to 0. The joint probability would then also be equal to zero, which is a problem if the probability is known or reasonably inferred to be non-zero.

In these cases, it is recommended to smooth the data as it is fed to the model. To smooth a data set is to perform a function on the data such that any undesired "noise" or localized phenomena are prevented from catastrophically influencing our model. When applied to the Naïve Bayes' model it may be defined as follows

$$\max_{v\,\in values(t)} P(t=v) \times \prod_{i=1}^{m} \frac{count(d[i] \mid t=v)+k}{count(d \mid t=v)+(k \times |Domain(d)|)})$$

With Laplace Smoothing applied, all calculated probabilities are guaranteed non-zero while still preserving their relative values, as changes to non-zero probabilities will be relatively tiny, and to zero probabilities, only significant enough to prevent model failure.

The Bayesian Network. But what do we do in cases where we would like to avoid computing joint probability tables AND holding any "naïve" assumptions? Another useful probabilistic model is the Bayesian network, which is particularly useful in its ability to account for conditionally dependent relationships without the computational complexity of joint probability distributions.

The Bayesian network model achieves this by encoding these structural relationships (i.e. direct influence and conditional independence) in the form of a tree-like, graphical representation

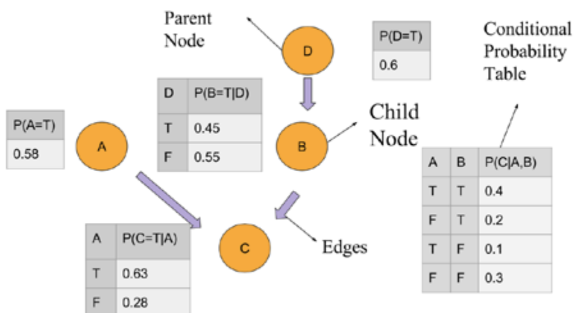


Figure 1. Bayesian Network

Each node in the figure above represents a feature, while each edge represents an influential relationship between the two nodes it connects. Next to each node is a table (i.e. a conditional probability table) containing the probability distribution across each of the possible values the corresponding feature given the conditioned of its parents. Using this information and Bayes' Theorem, the probability of any given set of feature-value assignments across the network may be calculated as follows

$$P(d[1],\ldots,d[m]) = \prod_{i=1}^{m} P(d[i] |\, Parents(d[i]))$$

As far as algorithms go, this is simple compared to others we have covered here. We iterate through each feature and calculate its probability conditioned on the state of its parents (Parents(d[i]) just returns the parents of d[i]. We then multiply each of these probabilities to calculate the corresponding joint probability.

It may be easier to understand how relationships of conditional dependence are encoded in our Bayesian network model if we consider an example of a Markov blanket: the nodes in a graph that make a particular node conditionally independent of others

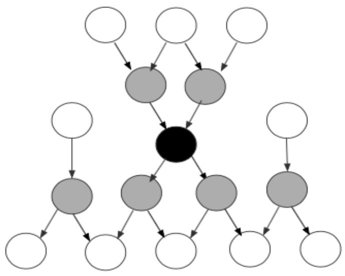


Figure 2. Markov blanket

Calculating the conditional probability of a target feature in a network requires knowledge of the states of all nodes that may inform our prediction, which includes not only its parents, but also the states of its children and each of their parents.

$$P(d[i] \mid d[1],\dots,d[i-1],d[i+1],\dots,d[m]) = P(d[i] \mid Parents(d[i])) \times \prod_{j \in Children(d[i])} P(d[j] \mid Parents(d[j]))$$

Like Parents(d[i]), Children(d[i]) is just the set containing the children of d[i]. This formula places the conditional probability of d[i] in terms of only a limited subset of the nodes contained by the Bayesian network. For Naïve Bayes' model's solution for the same problem, it looks very similar to our network's solution when reformulated like so

$$\max\left(P(t \mid Parents(t)) \times \prod_{j \in Children(t)} P(d[j] \mid Parents(d[j])) \right)$$

We just apply the original formula to our target feature, t, and return the value determined to be most probable.

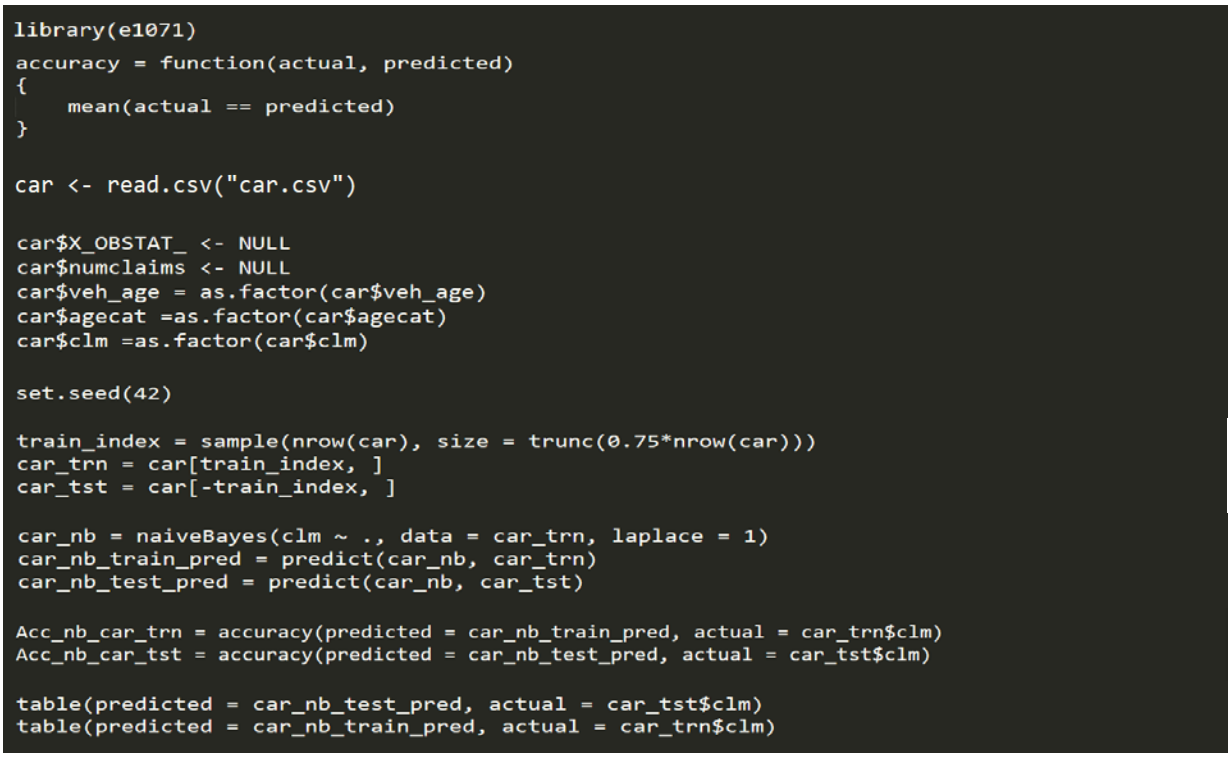
Solutions

Data Preparation. Before going into the implementation examples, we must first understand the data that will be manipulated. We will be using a data set from Piet de Jong's, "Generalized Linear Models for Insurance Data". The data set is based on one-year vehicle insurance policies taken out in 2004 and 2005 and provides values for ten variables describing the policy and policy owner. Piet de Jong's variable descriptions are as follows

Variable Name	Descriptions
veh_value	The value of vehicle in $10,000s
clm	The occurrence of claim (0 = no, 1 = yes)
numclaims	The number of claims
claimcst0	The claim amount (0 if no claim)
veh_body	vehicle body, coded as `BUS`, `CONVT` = convertible, `COUPE`, `HBACK` = hatchback, `HDTOP` = hardtop, `MCARA` = motorized caravan, `MIBUS` = minibus, `PANVN` = panel van, `RDSTR` = roadster, `SEDAN`, `STNWG` = station wagon, `TRUCK`, `UTE` = utility
veh_age	The age of vehicle: 1 (youngest), 2, 3, 4
gender	The gender of driver: M (male), F (female)
area	The driver's area of residence: A, B, C, D, E, F
agecat	The driver's age category: 1 (youngest), 2, 3, 4, 5, 6

Our target feature will be the "clm" and our descriptive features will be selected according to the model.

Naïve Bayes' Model Implementation. We chose to use the R programming language for our models, as R is a widely used open-source programming language equipped with multiple statistical models writing in its libraries. For Naïve Bayes' Model construction, we first download library named "e1071", which contains function for Naïve Bayes' Model, then we follow the preparation steps by defining a new function, "accuracy", which compares the actual values taken by a target feature in our dataset with those predicted by our model. It returns the ratio of correct predictions to total predictions.



```
library(e1071)
accuracy = function(actual, predicted)
{
    mean(actual == predicted)
}

car <- read.csv("car.csv")

car$X_OBSTAT_ <- NULL
car$numclaims <- NULL
car$veh_age = as.factor(car$veh_age)
car$agecat =as.factor(car$agecat)
car$clm =as.factor(car$clm)

set.seed(42)

train_index = sample(nrow(car), size = trunc(0.75*nrow(car)))
car_trn = car[train_index, ]
car_tst = car[-train_index, ]

car_nb = naiveBayes(clm ~ ., data = car_trn, laplace = 1)
car_nb_train_pred = predict(car_nb, car_trn)
car_nb_test_pred = predict(car_nb, car_tst)

Acc_nb_car_trn = accuracy(predicted = car_nb_train_pred, actual = car_trn$clm)
Acc_nb_car_tst = accuracy(predicted = car_nb_test_pred, actual = car_tst$clm)

table(predicted = car_nb_test_pred, actual = car_tst$clm)
table(predicted = car_nb_train_pred, actual = car_trn$clm)
```

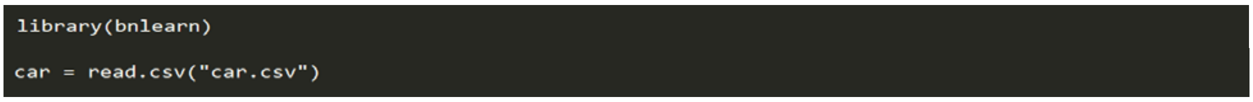
After that, we can read in the car insurance data. However, the data has to be cleaned before fitting the model. Clean data means all unrelated information are removed from the data. For example, in this case our target is to predict whether there is a claim or not, the number of claims happened does not matter to the occurrence of claim. Hence it should be removed. Also, there are some factor variables like age categories, occurrence of claims, ages of vehicles. We have to factorize them before fitting the model; otherwise R will take them as other ordinary numeric variables, which will affect the accuracy of our prediction.

After obtaining a set of clean data, we are able to split out 3/4 of the observations into the train group and remaining fall into the test group. We try to maximize the train group to ensure that we fit the Naïve Bayes' Model to as much data as possible, but we still want to leave a small portion of the data untouched for accuracy examination. However, before splitting the data we need to set a seed here. This ensures the reproduction of the same randomization of data set with fixed seed.

The Naïve Bayes' Model with Laplace smoothing of 1 gives us an accuracy of 93.16%. With that we are pretty confident with our prediction on occurrence of car insurance claims.

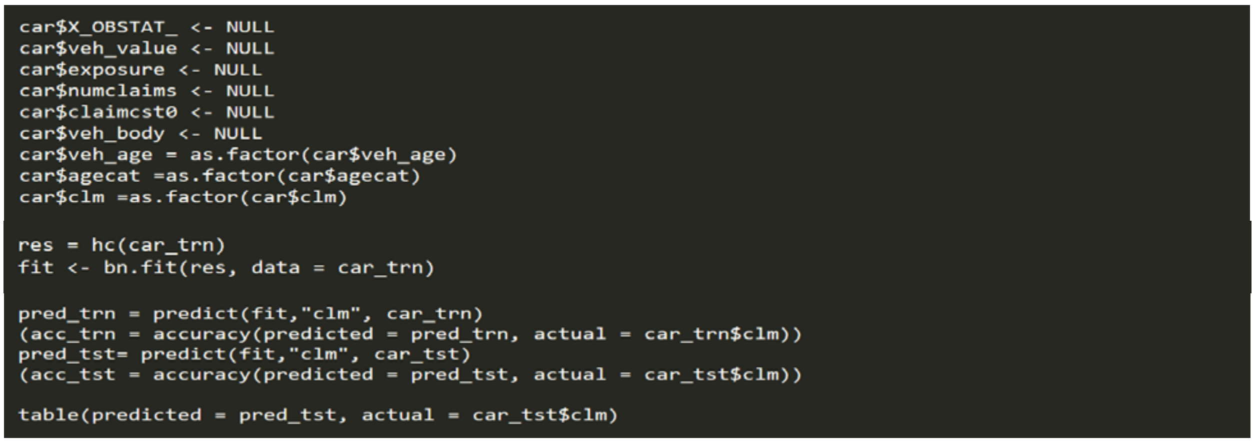
Bayesian Network Implementation. For Bayesian Network, we basically repeat the same steps as we did for the Naïve Bayes' Model, except the library installed this time is "bnlearn". One limitation of Bayesian Network is that it can only handle the same kind of variables that the dependent variable is. In this case, since 'clm' is a factor variable, we have to omit those continuous variables and then factorize factor variables.

After fitting Bayesian Network model to the clean and split data, we obtained an accuracy of 93.16%.



```
library(bnlearn)

car = read.csv("car.csv")
```



```
car$X_OBSTAT_ <- NULL
car$veh_value <- NULL
car$exposure <- NULL
car$numclaims <- NULL
car$claimcst0 <- NULL
car$veh_body <- NULL
car$veh_age = as.factor(car$veh_age)
car$agecat =as.factor(car$agecat)
car$clm =as.factor(car$clm)

res = hc(car_trn)
fit <- bn.fit(res, data = car_trn)

pred_trn = predict(fit,"clm", car_trn)
(acc_trn = accuracy(predicted = pred_trn, actual = car_trn$clm))
pred_tst= predict(fit,"clm", car_tst)
(acc_tst = accuracy(predicted = pred_tst, actual = car_tst$clm))

table(predicted = pred_tst, actual = car_tst$clm)
```

Conclusion

Both the Naïve Bayes' Model and the Bayesian Network model have reached the accuracy of 93.16%. Therefore we are very confident with the results both models get.

However, we do need to take Bayesian Network model's limitation into consideration. Since Bayesian Network model can only take in one kind of variable, there is probably a significant amount of data that does not applicable to Bayesian Network model. Therefore it is not as convenient as the Naïve Bayes' model. But in other hand, since Bayesian Network model arrives the same accuracy as the Naïve Bayes' model does with fewer variables, when coming into pricing insurance cases with limited information available, Bayesian Network model can be more effective.

References

[1] Uddin M F, Lee J. Proposing stochastic probability-based math model and algorithms utilizing social networking and academic data for good fit students prediction[J]. Social Network Analysis and Mining, 2017, 7(1): 29.

[2] Xu Q, Wang L, Wang N, et al. A review of opposition-based learning from 2005 to 2012[J]. Engineering Applications of Artificial Intelligence, 2014, 29: 1-12.

[3] Sheen S, Anitha R, Natarajan V. Android based malware detection using a multifeature collaborative decision fusion approach[J]. Neurocomputing, 2015, 151: 905-912.

[4] Fukuda S, Yamanaka Y, Yoshihiro T. A Probability-based Evolutionary Algorithm with Mutations to Learn Bayesian Networks[J]. IJIMAI, 2014, 3(1): 7-13.