R Project

Achyuth Pilly

NAÏVE BAYES CLASSIFIER

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any of it’s other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter.

Bayes Theorem

- Naïve Bayes Classifier is based on the Bayes theorem.
- Bayes' theorem can be used to make prediction based on prior knowledge and current evidence, with accumulating evidence, the prediction is changed.
- Bayes' theorem is formally expressed by the following equation.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

- where P(A) and P(B) are probability of events A and B without regarding each other. P(A|B) is the probability of A conditional on B and P(B|A) is the probability of B conditional on A. In naïve Bayes classification, A is categorical outcome events and B is a series of predictors. The word "naïve" indicates that the predictors are independent on each other conditional on the same outcome value.

Approach

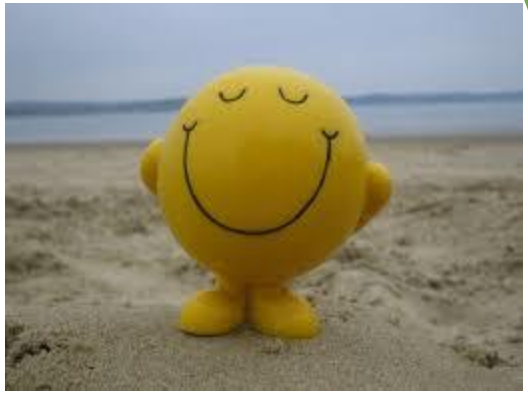
- Classifying the admissions of students into a University based on their GPA, GRE and the rank of the college/University from which they have graduated.
- Data has 4 columns with 400 observations:
  - Rank: Rank of the college/University.
  - GRE: GRE score of the student.
  - GPA: GPA acquired by the student.
  - Admit: categorical column with 0: Not admitted or 1: Admitted in to the University.

Dataset

admit	gre	gpa	rank
0	380	3.61	3
1	660	3.67	3
1	800	4	1
1	640	3.19	4
0	520	2.93	4
1	760	3	2
1	560	2.98	1
0	400	3.08	2
1	540	3.39	3
0	700	3.92	2
0	800	4	4
0	440	3.22	1
1	760	4	1
0	700	3.08	2
1	700	4	1
0	480	3.44	3
0	780	3.87	4
0	360	2.56	3
0	800	3.75	2
1	540	3.81	1
0	500	3.17	3
1	660	3.63	2
0	600	2.82	4
0	680	3.19	4
1	760	3.35	2
1	800	3.66	1
1	620	3.61	1
1	520	3.74	4
1	780	3.22	2
0	520	3.29	1
0	540	3.78	4
0	760	3.35	3
0	600	3.4	3
1	800	4	3
0	360	3.14	1
0	400	3.05	2
0	580	3.25	1

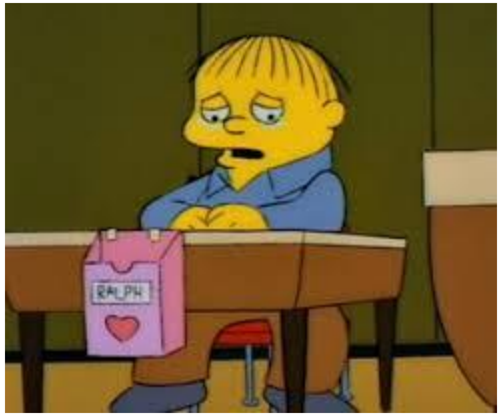
Student_admission

Pros of Naïve Bayes classifier



- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable. For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons of Naïve Bayes classifier



- If categorical variable has a category, which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as “Zero Frequency”. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- Naïve Bayes is also known as bad estimator.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Real time applications

Naive Bayes classifiers is a machine learning algorithm. Suppose, how Google marks some of the mails as spam in your inbox, a machine learning algorithm will be used to classify an incoming email as spam or not spam. Some of real world examples are as given below:

- To mark an email as spam, or not spam.
- Classify a news article about technology, politics, or sports.
- Check a piece of text expressing positive emotions, or negative emotions.
- Sentimental Analysis on twitter's tweets.

Code

Naive_bayes_prediction (1).R* | data

Source on Save

```r
# Libraries
install.packages("naivebayes")
install.packages("psych")
install.packages("dplyr")
install.packages("ggplot2")
library(naivebayes)
library(dplyr)
library(ggplot2)
library(psych)

# Data
data <- read.csv(file.choose(), header = T)
View(data)
str(data)
xtabs(~admit+rank, data = data)
data$rank <- as.factor(data$rank)
data$admit <- as.factor(data$admit)

# Visualization
data %>%
  ggplot(aes(x=admit, y=gpa, fill = admit)) +
  geom_boxplot() +
  ggtitle("Box Plot")
data %>%
  ggplot(aes(x=admit, y=gre, fill = admit)) +
  geom_boxplot() +
  ggtitle("Box Plot")

data %>% ggplot(aes(x=gpa, fill = admit)) +
  geom_density(alpha=0.8, color= 'black') +
  ggtitle("Density Plot")
data %>% ggplot(aes(x=gre, fill = admit)) +
  geom_density(alpha=0.8, color= 'black') +
  ggtitle("Density Plot")

# Data Partition
set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.8, 0.2))
train <- data[ind == 1,]
test <- data[ind == 2,]
```

```r
# Data Partition
set.seed(1234)
ind <- sample(2, nrow(data), replace = T, prob = c(0.8, 0.2))
train <- data[ind == 1,]
test <- data[ind == 2,]

# Naive Bayes Model
model <- naive_bayes(admit ~ ., data = train, usekernel = T)
model

plot(model)

# Predict
p <- predict(model, train, type = 'prob')
head(cbind(p, train))

# Confusion Matrix - train data
p1 <- predict(model, train)
(tab1 <- table(p1, train$admit))
1 - sum(diag(tab1)) / sum(tab1)

# Confusion Matrix - test data
p2 <- predict(model, test)
(tab2 <- table(p2, test$admit))
1 - sum(diag(tab2)) / sum(tab2)
```


Thank You!