Allison Molitor

M.L. Learning Journal

Start Date: 02/12/2022

Most Recent Edit: 08/24/2023

Machine Learning Classification Models

Naïve Bayes Modeling

The Naïve Bayes classification algorithm is based on Baye's theorem. It assumes all features are conditionally independent. Because of this, Naïve Bayes is very computationally efficient and simple to implement. The algorithm is primarily designed for predicting probabilities when given linear relationships but does not necessarily assume linearity. It is meant to be run on categorical features.

- Specialized Implementations
  - ***Gaussian Naïve Bayes***
    - Assumes input features follow a normal distribution
    - Calculates the probability of a feature value given each class using mean and standard deviation
  - ***Multinomial Naïve Bayes***
    - Assumes features represent discrete counts or frequencies and follows a Multinomial distribution
      - EX: text classification, XYZ word appears x amount of times
    - Calculates probabilities using counts and also employs soothing techniques to handle unseen occurrences
  - ***Bernoulli Naïve Bayes***
    - Used when features are binary indicators
      - EX: XYZ word is present or absent in the given statement
    - Assumes Bernoulli distribution
  - ***Hybrid Naïve Bayes***
    - Good for when a dataset has continuous, discrete, and binary features.

- Mathematical Representation
  - $P(C|X) = (P(X|C) * P(C)) / P(X)$
    - $P(C|X)$ = Probability of class C given the features X
    - $P(X|C)$ = Probability of features X given class C
    - $P(C)$ = Prior probability of class C
    - $P(X)$ = Prior probability of features X

- Real-world example
    - Classifying customer feedback
- Mathematical Computation:

Positive reviews: 100
Negative reviews: 100

Word identifiers: ["great", "awesome", "bad", "terrible", "amazing", "awful"]

Given feedback: "This product was great and awesome!"

Equation Set 1:

P(positive|review) = P(great|positive) * P(awesome|positive) * P(positive)

P(negative|review) = P(great|negative) * P(awesome|negative) * P(negative)

To fill out the above equations, we need to gather the following values from the training data (Equation Set 2):

P(positive) = Number of positive reviews / Total number of reviews
P(negative) = Number of negative reviews / Total number of reviews
P(great|positive) = Number of positive reviews containing "great" / Number of positive reviews
P(awesome|positive) = Number of positive reviews containing "awesome" / Number of positive reviews
P(great|negative) = Number of negative reviews containing "great" / Number of negative reviews
P(awesome|negative) = Number of negative reviews containing "awesome" / Number of negative reviews

Assuming the training data has the following values:

Number of positive reviews: 100
Number of negative reviews: 100
Number of positive reviews containing "great": 80
Number of positive reviews containing "awesome": 90
Number of negative reviews containing "great": 30
Number of negative reviews containing "awesome": 10

These numbers inserted into Equation Set 2 are:

P(positive) = 100 / 200 = 0.5
P(negative) = 100 / 200 = 0.5
P(great|positive) = 80 / 100 = 0.8
P(awesome|positive) = 90 / 100 = 0.9
P(great|negative) = 30 / 100 = 0.3
P(awesome|negative) = 10 / 100 = 0.1

Now we can solve Equation Set 1:

P(positive|review) = P(great|positive) * P(awesome|positive) * P(positive) = 0.36

P(negative|review) = P(great|negative) * P(awesome|negative) * P(negative) = 0.015

Given that the probability of it being a positive review (0.36) is greater than the probability of it being a negative review (0.015), Naive Bayes would classify "This product was great and awesome!" as positive feedback.

- Primary Advantages
  - Simplicity
    - Straightforward
    - Simple mathematical approach
    - Easy to implement
  - Efficiency
    - Computationally efficiency
    - Low training times
    - Scales well
  - Handles high-dimensional data well
    - Performs well with large amount of features
  - Works well with categorical features

- Primary Disadvantages
  - Assumption of Feature Independence
    - Naïve Bayes assumes that features are conditionally independent, which may not hold true in real-world scenarios.
  - Sensitive to Feature Correlation
    - When features are highly correlated, NaïveBayes may give less accurate results.
  - Requires Sufficient Training Data
    - Relies on a sufficient amount of data for accurate estimation of probabilities.
  - Can be affected by the "Zero Frequency" Problem

        - If a feature has not been observed in the training data with a specific class, it will assign zero probability, leading to incorrect predictions.
            - Can be mitigated by Laplace smoothing

- Common uses
    - Text Classification/Document categorization
        - As seen above
    - Medical Diagnosis/Disease Prediction
        - With Naïve Bayes, one can produce the conditional probability of a given classification. This could be used to estimate the risk of developing a disease or predicting which disease is most probable, given certain symptoms
    - Risk assessment
        - Calculates probabilities of certain events happening. This can be useful in healthcare, investment portfolios, etc.
    - Market segmentation based on customer attributes
        - Given customer features, Naïve Bayes can predict the most probable market segmentation

Logistic Regression (LOGIT) Modeling

Similarly to NaïveBayes, Logistic regression also predicts probabilities given linear relationships. It also is computationally efficient and easy to implement. It is the most useful when given a small/medium dataset with a limited number of features. One difference is that LOGIT is used only for binary outcomes (1 or 0). Another difference is it naturally handles continuous (non categorical) features.

- Specialized Implementations
    - ***Multinomial Logistic Regression:***
        - When one wishes to generate probabilities that extend past binary
    - ***Ordinal Logistic Regression (PROBIT):***
        - Ordinal outcomes with ordered categories
            - Further explanation at the bottom
    - ***Regularized Logistic Regression:***
        - Regularizes model with Lasso or Ridge regularization.
        - Improves overfitting
        - Good for high dimensional data
    - ***Bayesian Logistic Regression:***
        - Applies Bayesian framework to the logistic regression model

- Mathematical Representation
  - $P(Y = 1|X) = 1 / (1 + e^{(-z)})$
    - $P(Y = 1|X)$ = the probability of the outcome Y being 1 given the input features X
    - z = the linear combination of the input features and their respective coefficients
  - $z = \beta_0 + \beta_1 * X$
    - B_0 = odds of outcome being 1 when all input features =0
    - B_1 = change in odds per one unit increase of input features
- Real-world example
  - Predicting if a customer works from home, given the amount of hours interacting on the internet

- Mathematical Computation

Avg hours interacting on internet (X)	Works from home (Y)
2	0
3	0
4	0
5	1
6	1
7	1

When Y=1 the customer works from home

Assume estimated coefficients from training process determine
$\beta_0 = -3$
$\beta_1 = 0.8$

Say we wish to calculate the odds of a customer working from home given they interact on the internet for 5 hours.

$z = \beta_0 + \beta_1 * X$
$z = -3 + 0.8 * 5$
$z = -3 + 4$
$z = 1$
Given z = 1, we can now solve

P(Y = 1|X) = 1 / (1 + e^(-z))
P(Y = 1| (5)) = 1 / (1 + e^(-(1)))
P(Y = 1| (5)) = 1 / (1 + e^(-1))
P(Y = 1| (5)) = 1 / (1 + 0.368)
P(Y = 1| (5)) = 1 / 1.368
P(Y = 1| (5)) = 0.73

Given the probability of Y=1 = .73, one can predict that if, on average, a customer uses the internet for 5 hours a day, there is a 73% chance the customer works from home. A network provider may capitalize on this by providing ads that are geared toward a remote worker. This capitalization could be offerings of higher network stability for Zoom calls or cloud networks for file transferring.

- Primary Advantages
  - Simplicity
    - Straightforward
    - Simple mathematical approach
    - Easy to implement
  - Efficiency
    - Computationally efficiency
    - Low training times
    - Scales well
  - Can Handle Continuous features
    - Suitable for datasets with different types of features
  - Handles complex relationships among features well
    - More flexible towards non-linear relationships
  - Handles imbalances in data well
    - Can adjust class weights to achieve maximum accuracy
- Primary Disadvantages
  - Prone to Overfitting
    - Especially when given large and or complex data sets
  - Limited to Binary Classifications
    - 1 or 0
  - Sensitive to Outliers

Proportional Odds Logistic Regression (PROBIT) Modeling

Similar to logistic regression, PROBIT modeling predicts probabilities based on linear relationships. It is also computationally efficient and well-suited for small to medium-sized data sets. PROBIT works best with a moderate number of features and is designed for ordinal outcomes. It assumes constant relationships across all ordinal outcomes.

- Example where this holds true:

Predicting performance on an exam based on number of hours studied. One would expect the relationship between hours studied and exam performance to be the same regardless of exam performance level (A, B, C, etc.)

- Example where this does not hold true:

Customer satisfaction of a service based on the age of a customer. The relationship across various ages (20-30, 30-40 , etc.) and customer satisfaction is not constant. This is because it is unclear how differing age brackets influence customer satisfaction, whereas we expect the influence of hours studied upon exam performance to be relatively the same regardless of the exam performance bracket.