Indian Institute of Technology Patna

CS571:AI and ML Lab

ASSIGNMENT-4: Naive Bayes
Submission Date: 26th April 2024

Baskar Natarajan - 2403res19(IITP001799)
Jyotisman Kar – 2403res35(IITP001751)
SEMESTER-1
MTECH AI & DSC

INDIAN INSTITUTE OF TECHNOLOGY PATNA.

Problem Description:

Problem Statement:

Given a dataset containing various details about the details of employees such as 'workclass,' 'education,' 'occupation' etc. The task is to predict the 'income' of the employee given the other details. The salary field contains two values '<= 50k' and '>50k'.

Dataset:
https://www.dropbox.com/scl/fi/nmf77m7tz82i71tyi9fjc/adult.csv?rlkey=pqv7w2lshmac2qx03lza7imxh&dl=0

Link Implementation Details:

- Implement two Naive Bayes classifiers:
  - Gaussian Naive Bayes
  - Multinomial Naive Bayes
- You can use existing packages such as Scikit-Learn to create the classifier
- Perform a 60-20-20 split on the dataset for train, validation, and test set.
- Perform 4-fold cross-validation.
- Report accuracy on the test set for both classifiers.
- Drop all null-valued columns.

Documents to submit:

- Model code
- Detailed document describing results such as confusion matrix for the execution and accuracy results

What is Join Probability?

- It describes the likelihood of different outcomes occurring together for multiple variables.
- In many real-world scenarios, there are multiple variables that can influence outcomes.
- For example, in a weather prediction model, variables such as temperature, humidity, and wind speed may all affect the likelihood of rain.

What is Generative Model?

- A generative machine learning model is a type of model that learns the joint probability distribution of the input features and the target labels.
- It learns the underlying data-generating process and can generate new samples that resemble the training data.
- Generative models are contrasted with discriminative models, which learn the conditional probability distribution of the target labels given the input features.
    - What is conditional probability distribution?
        - It describes how the probability of one variable is affected by the presence or absence of specific values of another variable. It quantifies the uncertainty in one variable when the value of another variable is known.
    - joint probability represents the likelihood of observing specific combinations of outcomes for multiple random variables, while conditional probability represents the likelihood of one event occurring given that another event has already occurred.
    - What is the discriminative model?
        - Discriminative models are a class of machine learning models that directly learn the mapping from input features to output labels without explicitly modeling the underlying probability distribution of the data.
        - These models focus on learning the decision boundary between different classes or categories in the input space. Discriminative models discriminate between different classes based on the input features.

What is Bayes' Theorem?

- Naive Bayes is a simple probabilistic classifier based on Bayes' theorem with the "naive" assumption of feature independence given the class.
- It is widely used in various machine learning applications, including text classification, spam filtering, recommendation systems, and medical diagnosis.

Bayes' Theorem:

- At the core of Naive Bayes is Bayes' theorem, which describes the probability of a hypothesis given the evidence:
    - A and B are two events.
    - $P(A)$ and $P(B)$ be the probabilities of events A and Boccurring, respectively.
    - $P(A|B)$ be the conditional probability of event A occurring given that event BB has occurred.
    - $P(B|A)$ be the conditional probability of event B occurring given that event A has occurred.

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

- In the context of classification, the theorem helps us compute the probability of a class label given the input features.

Naive bayes:

Naive Bayes makes the "naive" assumption that the features are conditionally independent given the class label.

This assumption simplifies the computation significantly, as it allows us to compute the probability of each feature independently.

$$P(H \setminus E) = \frac{P(E \setminus H) * P(H)}{P(E)}$$

A **hypothesis** is a tentative explanation or prediction about the relationship between variables or the underlying structure of data.

- P(H\E) -- Prior probability of the hypothesis given that the evidence is true.
- P(E) -- prior probability that evidence is true.
- P(H) -- Prior probability of the hypothesis

- P(E/H) -- Likelihood of the evidence given that the hypothesis is True.

For example, in our dataset:

- We need to find the income of the employees.
- **P(Income | Features) = P(Features|Income) X P(Income) / P(Features)**

Classification Process:

1. Given a set of features $x1$, $x2$..., xn and a class label y, Naive Bayes calculates the probability of each class given the features using Bayes' theorem.
2. The class with the highest probability given the features is then predicted as the output.

Types of Naive Bayes:

- Naive Bayes comes in different variants based on the distribution of the features:
- **Gaussian Naive Bayes**: Assumes that the features follow a Gaussian (normal) distribution.
- **Multinomial Naive Bayes**: Suitable for features that represent counts or frequencies, often used in text classification tasks.
- **Bernoulli Naive Bayes**: Assumes binary features, typically used for text classification with binary feature vectors.

Key Features:

- **Simple and fast**: Naive Bayes is computationally efficient and scales well to large datasets because of its simplicity.
- **Requires less training data**: It performs well even with a small amount of training data.
- **Handles categorical and continuous features**: Depending on the variant used, Naive Bayes can handle both categorical and continuous features.

When to Use:

- Naive Bayes is suitable for classification tasks, especially when the independence assumption holds well in the data.
- It is commonly used in text classification tasks, such as spam filtering, sentiment analysis, and document categorization, but it can be applied to various domains.

Summary:

- Naive Bayes is a simple yet effective probabilistic classifier that leverages Bayes' theorem and the naive assumption of feature independence to make predictions efficiently.

- Despite its simplicity, it is often used in practice and can perform well in a wide range of applications.

Gaussian Naive Bayes:

- It is a variant of the Naive Bayes algorithm specifically designed for classification tasks with continuous or real-valued input features.
- It assumes that the likelihood of the features given the class label follows a Gaussian (normal) distribution

How Gaussian Naive Bayes works?

- It calculates the likelihood of observing a particular value for each feature given the class label, assuming that the feature values are normally distributed within each class.
- By applying Bayes' theorem and the assumption of feature independence, it calculates the posterior probability of each class label given the observed features and selects the class label with the highest probability as the predicted label.

When to use Gaussian Naive Bayes?

- It is for classification tasks where the input features are continuous or real-valued and the assumption of feature independence holds well.
- It is particularly useful when the dataset is small, and computational efficiency is important, as Gaussian Naive Bayes is computationally efficient and scales well to large datasets.

Example related to predicting employee income (Current Problem):

- Suppose we have a dataset containing various details about employees, such as age, education level, work experience, and hours worked per week, along with their corresponding income levels (<=50k or >50k).
- We can use Gaussian Naive Bayes to predict the income level of employees based on these features.
- For each class label (income level), Gaussian Naive Bayes would estimate the mean and standard deviation of each continuous feature (e.g., age, hours worked per week) within that class.
- When making predictions for a new employee, Gaussian Naive Bayes would calculate the likelihood of observing their feature values given each class label using the Gaussian probability density function.
- Finally, it would select the class label with the highest posterior probability as the predicted income level for the employee.

Multinomial Naive Bayes:

- It is another variant of the Naive Bayes algorithm, specifically designed for classification tasks with discrete features, typically representing counts or frequencies.
- It is commonly used in text classification tasks where the features are often represented as word counts or term frequencies.

How does Multinomial Naive Bayes work?

- Multinomial Naive Bayes assumes that the features (variables) are generated from a multinomial distribution, which is suitable for discrete features representing counts or frequencies.
- It calculates the likelihood of observing a particular count or frequency for each feature given the class label.
- By applying Bayes' theorem and the assumption of feature independence, it calculates the posterior probability of each class label given the observed features and selects the class label with the highest probability as the predicted label.

When to use Multinomial Naive Bayes?

- Multinomial Naive Bayes is suitable for classification tasks where the features are discrete and represent counts or frequencies, such as in text classification tasks with bag-of-words or TF-IDF representations.
- It is commonly used in text classification, spam filtering, sentiment analysis, and other tasks involving categorical or count-based features.
- Multinomial Naive Bayes works well with datasets containing many features and is computationally efficient.

Example related to predicting employee income (Current Problem):

- Suppose we have a dataset containing various details about employees, including categorical features such as education level (e.g., high school, bachelor's, master's), occupation (e.g., management, sales, service), and workclass (e.g., private, government, self-employed).
- We can use Multinomial Naive Bayes to predict the income level of employees based on these categorical features.
- For each class label (income level), Multinomial Naive Bayes would estimate the probability of each category for each categorical feature within that class.
- When making predictions for a new employee, Multinomial Naive Bayes would calculate the likelihood of observing their feature values given each class label using the multinomial probability distribution.

- Finally, it would select the class label with the highest posterior probability as the predicted income level for the employee.

Compare Gaussian Naive Bayes and Multinomial Naive Bayes:

Feature Representation:

- GNB: Assumes features follow a Gaussian (normal) distribution, suitable for continuous or real-valued features like age and hours worked per week.
- MNB: Assumes features are generated from a multinomial distribution, suitable for discrete features representing counts or frequencies like education level, occupation, and workclass.

Feature Independence Assumption:

- GNB: Assumes feature independence given the class label, which may not hold true for all continuous features but is often a reasonable approximation.
- MNB: Also assumes feature independence given the class label, which may hold true for discrete features like categorical variables.

Performance:

- GNB: May perform well when the Gaussian assumption holds, and the features are normally distributed. However, it may not perform well if features are highly skewed or do not follow a Gaussian distribution.
- MNB: Tends to perform well in text classification tasks where the feature representation is sparse, and the assumption of feature independence holds well.

Computational Complexity:

- GNB: Involves estimating the mean and variance for each feature within each class, which can be computationally expensive for high-dimensional datasets.
- MNB: Involves counting the occurrences of each feature value within each class, which is computationally efficient and scales well to large datasets with sparse feature representations.

From Our example:

In the context of predicting employee income, if the dataset contains continuous features like age and hours worked per week, Gaussian Naive Bayes might be more suitable. On the other hand, if the dataset contains mostly discrete features like education level and occupation, Multinomial Naive Bayes could be a better choice.

Conclusion:

The choice between Gaussian and Multinomial Naive Bayes depends on the nature of the features in the dataset and the underlying assumptions of each algorithm.

Model using Gaussian and Multinomial Naive Bayes Classifiers

Description:

- This Python program compares the performance of Gaussian Naive Bayes and Multinomial Naive Bayes classifiers on a given dataset.
- It utilizes the Adult dataset, which contains demographic information about individuals and their income levels.
- The program incorporates label encoding for categorical variables, splits the dataset into training and testing sets,
- trains both classifiers, evaluates their accuracies, performs cross-validation, and plots confusion matrices to visualize the performance.

Dependencies:

- Pandas – to load CSV data and handle data frame operations
- scikit-learn – to plot the graph
- Matplotlib – plot the graph
- Seaborn – preprocessing and algorithms

Usage:

- Ensure that the "adult.csv" dataset is present in the same directory as the Python script.
- Run the script. It will load the dataset, preprocess it by encoding categorical variables,
- Split it into training and testing sets,
- Train Gaussian Naive Bayes and Multinomial Naive Bayes classifiers,
- Evaluate their accuracies,
- Compare the accuracies,
- Perform cross-validation,
- and plot confusion matrices.

Functionality:

Data Loading and Preprocessing:

- The program loads the "adult.csv" dataset and drops any null-valued columns.
- Categorical variables are encoded using label encoding from scikit-learn's preprocessing module (only for Gaussian model).
- The Multinomial Naive Bayes classifier (mnb) now uses CountVectorizer to convert the text data into word frequency vectors.
- The text features are concatenated to form a single string and then transformed using CountVectorizer.
    - **What is CountVectorizer and why required?**
        - It is a text processing technique that converts a collection of text documents into a matrix of token counts
        - we use CountVectorizer in our program to preprocess textual categorical features and convert them into a format suitable for the Multinomial Naive Bayes classifier, enabling effective text classification and improving the overall performance of the model.

Model Training and Evaluation:

- The dataset is split into features (X) and the target variable (y).
- The data is further split into training and testing sets using a 60-20-20 split ratio.
- Gaussian Naive Bayes and Multinomial Naive Bayes classifiers are initialized and trained on the training data.
- The accuracy of each classifier is evaluated on the testing set.

Comparison of Accuracies:

- The program compares the accuracies of Gaussian Naive Bayes and Multinomial Naive Bayes classifiers and prints which classifier has a higher accuracy.

Cross-Validation:

- 4-fold cross-validation is performed for both classifiers on the training and validation sets.
- Cross-validation scores are printed to assess the generalization performance of the classifiers.

Confusion Matrix Visualization:

- Confusion matrices are computed for both classifiers using the testing set.
- The confusion matrices are plotted using seaborn's heatmap for visualization.

Output:

- The program outputs the accuracies of Gaussian Naive Bayes and Multinomial Naive Bayes classifiers.
- It also displays which classifier has a higher accuracy or if they have the same accuracy.
- Cross-validation scores for both classifiers are printed.
- Confusion matrices are visualized for both classifiers, allowing for an assessment of their performance in terms of true positives, true negatives, false positives, and false negatives.

One of the Output:

Gaussian Naive Bayes Accuracy: 0.7978091728091729

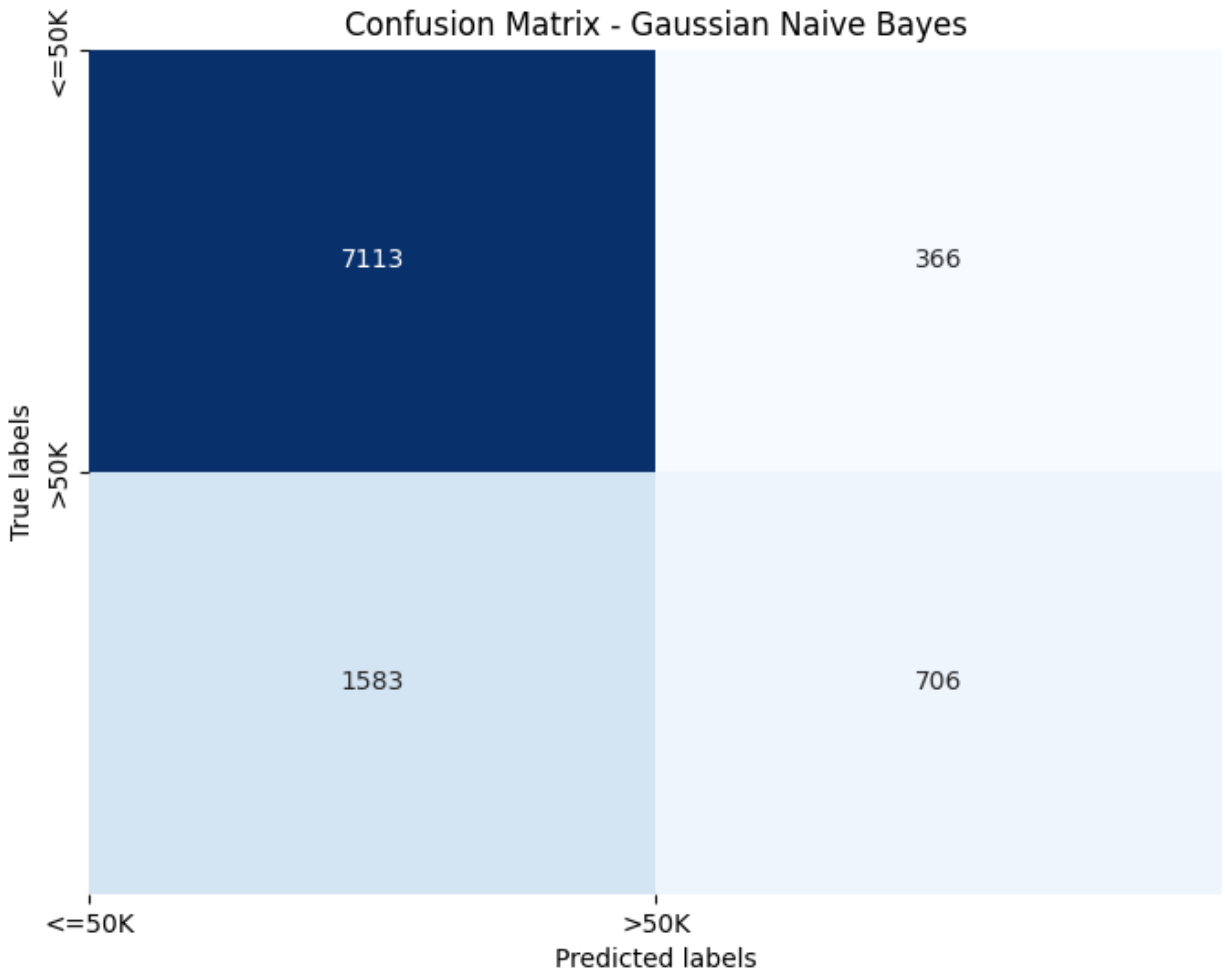
Multinomial Naive Bayes Accuracy: 0.7901310401310402

Gaussian Naive Bayes Cross-Validation Scores: [0.7968059 0.79328419 0.7968878 0.79356371]

Multinomial Naive Bayes Cross-Validation Scores: [0.78673219 0.78239148 0.78443898 0.78013429]

Gaussian Naive Bayes has higher accuracy.

Confusion Matrix – Gaussian Naive Bayes



Confusion Matrix – Multinomial Naive Bayes