Classification Algorithms - Naïve Bayes

Introduction to Naïve Bayes Algorithm

Naïve Bayes algorithms is a classification technique based on applying Bayes' theorem with a strong assumption that all the predictors are independent to each other. In simple words, the assumption is that the presence of a feature in a class is independent to the presence of any other feature in the same class. For example, a phone may be considered as smart if it is having touch screen, internet facility, good camera etc. Though all these features are dependent on each other, they contribute independently to the probability of that the phone is a smart phone.

In Bayesian classification, the main interest is to find the posterior probabilities i.e. the probability of a label given some observed features, $P(L \mid features)$. With the help of Bayes theorem, we can express this in quantitative form as follows −

$$P(L|features) = \frac{P(L)P(features|L)}{P(features)}$$

Here, $(L \mid features)$ is the posterior probability of class.

$P(L)$ is the prior probability of class.

$P(features|L)$ is the likelihood which is the probability of predictor given class.

$P(features)$ is the prior probability of predictor.

Building model using Naïve Bayes in Python

Python library, Scikit learn is the most useful library that helps us to build a Naïve Bayes model in Python. We have the following three types of Naïve Bayes model under Scikit learn Python library −

Gaussian Naïve Bayes

It is the simplest Naïve Bayes classifier having the assumption that the data from each label is drawn from a simple Gaussian distribution.

Multinomial Naïve Bayes

Another useful Naïve Bayes classifier is Multinomial Naïve Bayes in which the features are assumed to be drawn from a simple Multinomial distribution. Such kind of Naïve Bayes are most appropriate for the features that represents discrete counts.

Bernoulli Naïve Bayes

Another important model is Bernoulli Naïve Bayes in which features are assumed to be binary (0s and 1s). Text classification with 'bag of words' model can be an application of Bernoulli Naïve Bayes.

Example

Depending on our data set, we can choose any of the Naïve Bayes model explained above. Here, we are implementing Gaussian Naïve Bayes model in Python −

We will start with required imports as follows −

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
```

Now, by using *make_blobs()* function of *Scikit learn*, we can generate blobs of points with Gaussian distribution as follows −

```
from sklearn.datasets import make_blobs
X, y = make_blobs(300, 2, centers = 2, random_state = 2, cluster_std = 1.5)
plt.scatter(X[:, 0], X[:, 1], c = y, s = 50, cmap = 'summer');
```

Next, for using *GaussianNB* model, we need to import and make its object as follows −

```
from sklearn.naive_bayes import GaussianNB
model_GBN = GaussianNB()
model_GNB.fit(X, y);
```

Now, we have to do prediction. It can be done after generating some new data as follows −

```
rng = np.random.RandomState(0)
Xnew = [-6, -14] + [14, 18] * rng.rand(2000, 2)
ynew = model_GNB.predict(Xnew)
```

Next, we are plotting new data to find its boundaries −

```
plt.scatter(X[:, 0], X[:, 1], c = y, s = 50, cmap = 'summer')
lim = plt.axis()
plt.scatter(Xnew[:, 0], Xnew[:, 1], c = ynew, s = 20, cmap = 'summer', alpha = 0.1)
plt.axis(lim);
```

Now, with the help of following line of codes, we can find the posterior probabilities of first and second label −

```
yprob = model_GNB.predict_proba(Xnew)
yprob[-10:].round(3)
```

Output

```
array([[0.998, 0.002],
   [1. , 0. ],
   [0.987, 0.013],
   [1. , 0. ],
   [1. , 0. ],
   [1. , 0. ],
   [1. , 0. ],
   [1. , 0. ],
   [0. , 1. ],
   [0.986, 0.014]])
```

Pros & Cons

Pros

The followings are some pros of using Naïve Bayes classifiers −

- Naïve Bayes classification is easy to implement and fast.
- It will converge faster than discriminative models like logistic regression.
- It requires less training data.
- It is highly scalable in nature, or they scale linearly with the number of predictors and data points.
- It can make probabilistic predictions and can handle continuous as well as discrete data.
- Naïve Bayes classification algorithm can be used for binary as well as multi-class classification problems both.

Cons

The followings are some cons of using Naïve Bayes classifiers −

- One of the most important cons of Naïve Bayes classification is its strong feature independence because in real life it is almost impossible to have a set of features which are completely independent of each other.
- Another issue with Naïve Bayes classification is its ‘zero frequency’ which means that if a categorial variable has a category but not being observed in training data set, then Naïve Bayes model will assign a zero probability to it and it will be unable to make a prediction.

Applications of Naïve Bayes classification

The following are some common applications of Naïve Bayes classification −

- **Real-time prediction** − Due to its ease of implementation and fast computation, it can be used to do prediction in real-time.
- **Multi-class prediction** − Naïve Bayes classification algorithm can be used to predict posterior probability of multiple classes of target variable.
- **Text classification** − Due to the feature of multi-class prediction, Naïve Bayes classification algorithms are well suited for text classification. That is why it is also used to solve problems like spam-filtering and sentiment analysis.
- **Recommendation system** − Along with the algorithms like collaborative filtering, Naïve Bayes makes a Recommendation system which can be used to filter unseen information and to predict weather a user would like the given resource or not.