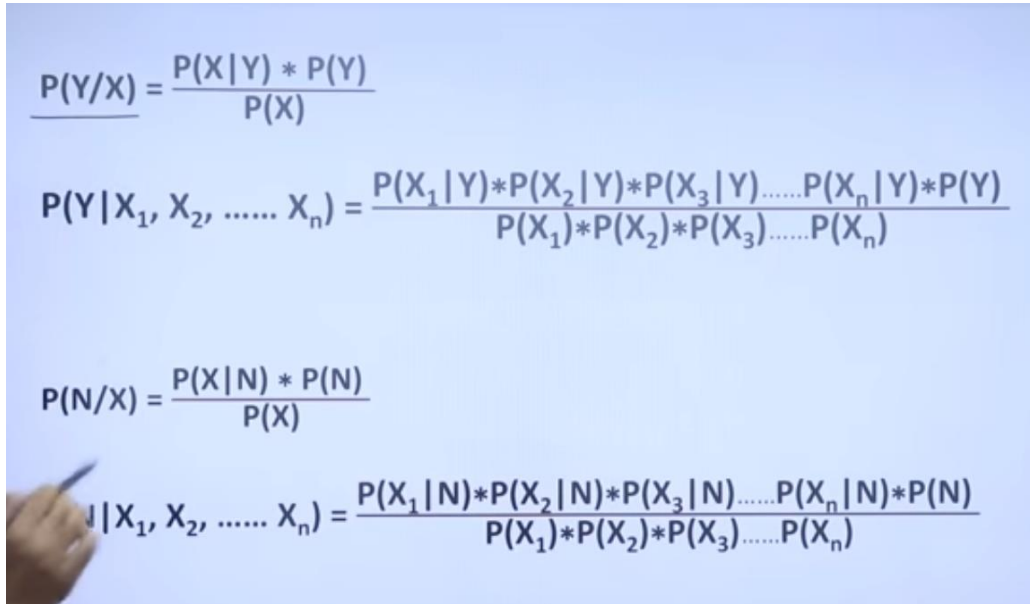
Naïve bayes for more ind variables



Example

Person	COVID (Yes/No)	Flu (Yes/No)	Fever (Yes/No)
1	Yes	No	Yes
2	No	Yes	Yes
3	Yes	Yes	Yes
4	No	No	No
5	Yes	No	Yes
6	No	No	Yes
7	Yes	No	Yes
8	Yes	No	No
9	No	Yes	Yes
10	No	Yes	No

Step-1: Prior Probability:

P(fever = yes) = 7/10

P(fever = no) = 3/10

Step-2: Conditional Probability:

	Yes	No
Covid	4/7	2/3
Flu	3/7	2/3

Given Person(flu, Covid)

P(yes| flu, Covid)= P(flu|yes)*P(covid|yes)* P(yes)

P(NO|flu, Covid)= P(flu|NO)*P(covid|No)*P(NO)

Advantages of Naive Bayes Classifier

- The following are some of the benefits of the Naive Bayes classifier:
- It is simple and easy to implement
- It doesn’t require as much training data
- It handles both continuous and discrete data
- It is highly scalable with the number of predictors and data points
- It is fast and can be used to make real-time predictions

Lab activity

- We load the Iris dataset from scikit-learn.
- Split the dataset into training and testing sets.
- Initialize a Gaussian Naive Bayes classifier (GaussianNB).
- Train the classifier using the training data.
- Make predictions on the testing data using the trained classifier.
- Evaluate the accuracy of the classifier by comparing the predicted labels with the actual labels from the testing set.

Iris dataset from scikit-learn

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
```

Separating dep and ind variables and splitting data

```
X = iris.data
y = iris.target
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Initialize the Gaussian Naive Bayes classifier

```
nb_
```

```
# Initialize the Gaussian Naive Bayes classifier
nb_classifier = GaussianNB()
```

Train the classifier -training data.

nb_cla

```
[11] # Train the classifier on the training data
     nb_classifier.fit(X_train, y_train)
```

Make predictions -testing data using

y_pr

0s

```
# Make predictions on the testing data
y_pred = nb_classifier.predict(X_test)
```

Evaluation of model