Naïve Bayes Classification

Subham Roy
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260
subhamro@buffalo.edu

1 Overview

Classification is the task of grouping things together because of the similarity they share. It helps organize things and thus makes the study easier and more systematic. In statistics, classification refers to the problem of identifying to which set of categories an observation or data value belongs to. In this report, we will work using Naïve Bayes classifier and analyze the outcome of the experiment.

2 About Naïve Bayes Classifier

A simple and robust classifier that belongs to the family of probabilistic classifiers. It follows the idea of the Bayes Theorem assuming that every feature is independent of every other feature. In the 1770s, Thomas Bayes introduced the 'Bayes Theorem'. Even centuries later, the importance of 'Bayesian Statistics' hasn't faded away.

Bayes' theorem is stated mathematically as the following equation:

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

where:

A, B = Events

P(B ∣ A) = The probability of B occurring given that A is true.

P(A) and P(B) = The probabilities of A occurring and B occurring independently of each other. Also, P(A) and P(B) ≠ 0

Therefore, Posterior Probability is thus the resulting distribution, P(A|B).

Given the categorical features (not real-valued data) along with categorical class labels, Naive Bayes computes the likelihood for each category from every feature concerning each category of class labels. Thus, it will choose a specific category of a class label whose likelihood is maximum.

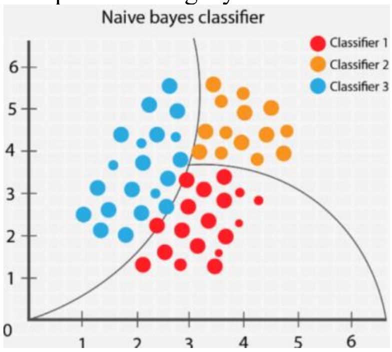


Fig: Plot of Naïve Bayes classifier

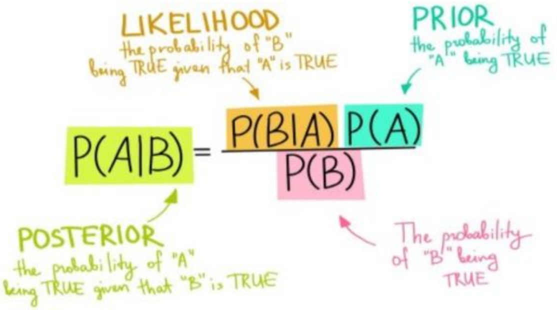

LIKELIHOOD
the probability of "B" being TRUE given that "A" is TRUE
PRIOR
the probability of "A" being TRUE
P(A|B) = P(B|A) P(A) / P(B)
POSTERIOR
the probability of "A" being TRUE given that "B" is TRUE
The probability of "B" being TRUE

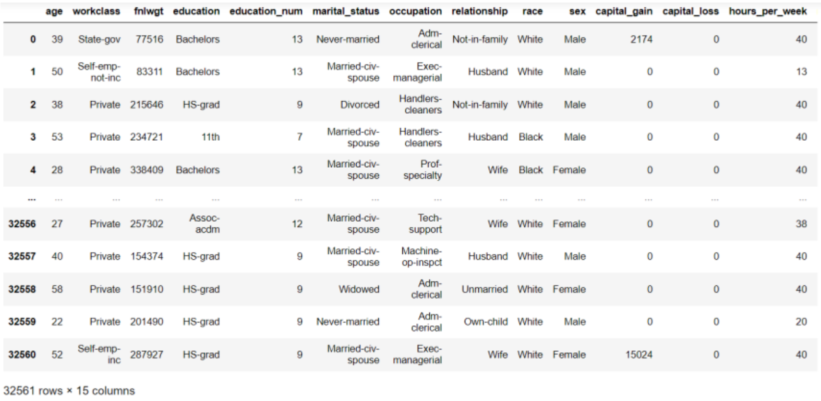
Fig: Bayes theorem formula

Naive Bayes is called naive because it assumes that each input variable is independent. This is a strong assumption and unrealistic for real data; however, the technique is very effective on a large range of complex problems.

3 Experiment

In this experiment, I used the ADULT.csv dataset from Kaggle. Initially, we start with loading the Dataset and checking the dimensions and the number of variables within the Dataset. In this case, the dataset has 15 features and 32561 rows.

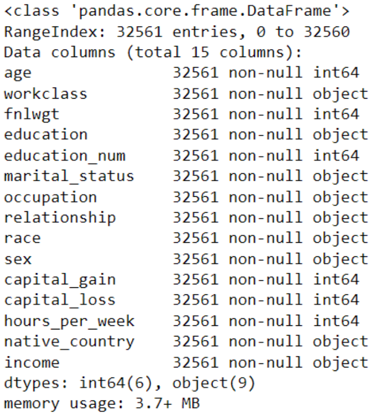
Dataset description –



	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40
...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40

32561 rows × 15 columns

Fig: Dataset

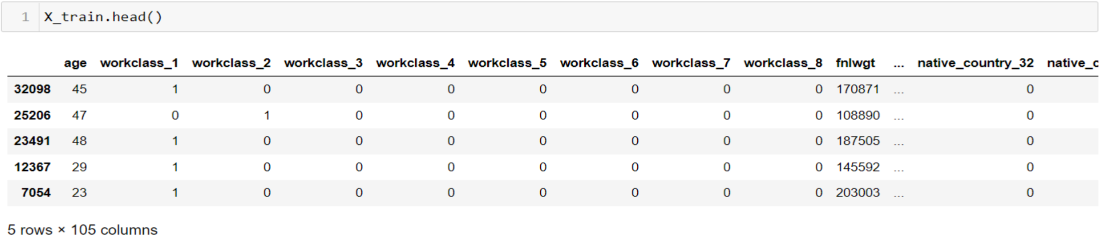


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
age               32561 non-null int64
workclass         32561 non-null object
fnlwgt            32561 non-null int64
education         32561 non-null object
education_num     32561 non-null int64
marital_status    32561 non-null object
occupation        32561 non-null object
relationship      32561 non-null object
race              32561 non-null object
sex               32561 non-null object
capital_gain      32561 non-null int64
capital_loss      32561 non-null int64
hours_per_week    32561 non-null int64
native_country    32561 non-null object
income            32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Fig: Dataset Column Datatypes

The next step is we perform data preprocessing. We perform the following steps before we can be certain that we can apply the classifier model. This is almost similar to all the techniques we apply. First, we check for missing data, then check for the spread of the data like where the Mean, Median of the data, are there any outliers, and we take care of that.

In this dataset, categorical variables are given by workclass, education, marital_status, occupation, relationship, race, sex, native_country, and income. Here, Income is the target variable. In these features, there are no Null values. Rest columns/features had some Null values which were in the form of '?'. Initially, we detect these values and then convert these to NaN, so that python treats them as Null values in the future. We now split the dataset into the Train set and Test set with a ratio of 80:20. After the split, we replace the Null values with the Mean of the individual columns, so that Train data values do not impact Test Data values. We also select all the categorical columns and perform One-Hot encoding to transform Object values to Binary.



```
X_train.head()
```

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_6	workclass_7	workclass_8	fnlwgt	...	native_country_32	native_c
32098	45	1	0	0	0	0	0	0	0	170871	...	0	
25206	47	0	1	0	0	0	0	0	0	108890	...	0	
23491	48	1	0	0	0	0	0	0	0	187505	...	0	
12367	29	1	0	0	0	0	0	0	0	145592	...	0	
7054	23	1	0	0	0	0	0	0	0	203003	...	0	

5 rows × 105 columns

Fig: Dataset after OneHotEncoding

Lastly, we scale the Train set and Test set separately so that one particular column's high variance does not highly influence the model prediction. Now we implement the Gaussian Naïve Bayes model on the Training set.

4 Analysis

We find out a lot of information about our prediction from this model. Firstly, we set a Classification threshold level of 0.5 as a baseline -

- Class 0 => <=50K – the probability of salary less than or equal to 50K is predicted if probability < 0.5.
- Class 1 => >50K – the probability of a salary of more than 50K is predicted if probability > 0.5.

The model accuracy for all the predictions is close to 81%, and the Null accuracy score of 75%. We can infer from this information that Gaussian Naïve Bayes is doing a pretty good job with this particular training set. Also, similar results are reflected when the model is fitted on Test data. That means there are no signs of overfitting. Using the confusion matrix, we analyze the errors in Classification –

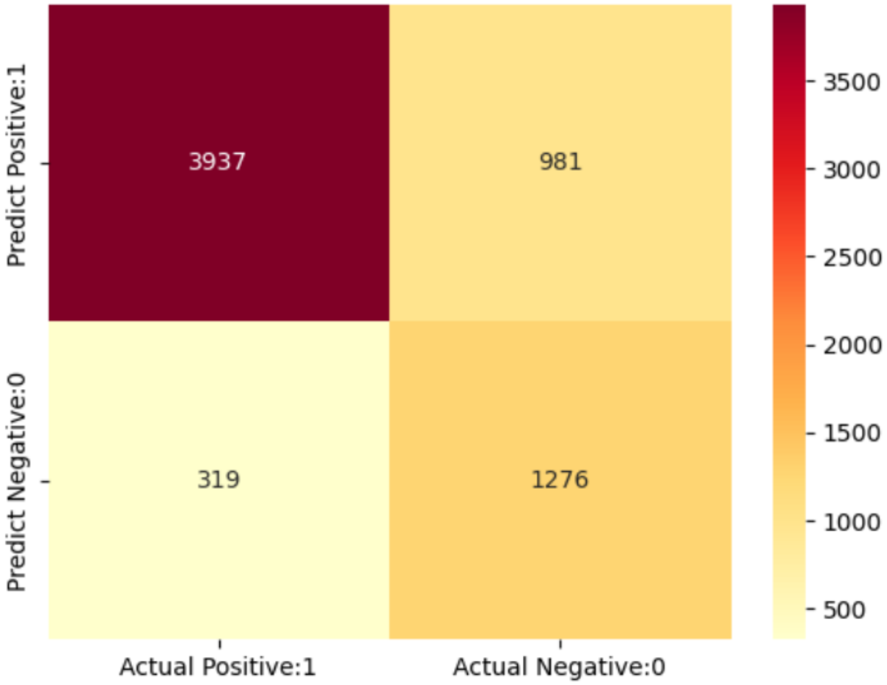


Fig: Confusion Matrix

The Model had a classification accuracy of 80%, a True positive rate of 92.5%, and a False positive of 43%. Another tool we used to measure the classification model performance visually was the ROC Curve. ROC Curve stands for Receiver Operating Characteristic Curve. A ROC Curve is a plot that shows the performance of a classification model at various classification threshold levels.

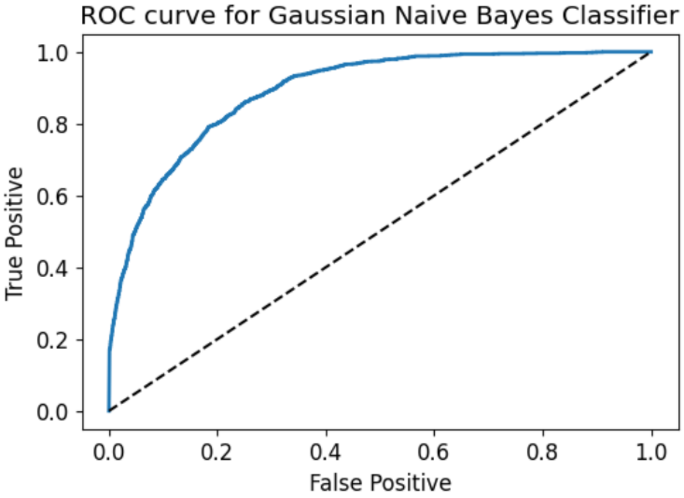


Fig: ROC Curve

References

[1] Gareth James [at], Daniela Witten [aut], Trevor Hastie [aut, cre], Rob Tibshirani [aut], Balasubramanian Narasimhan [ctb], *Introduction to Statistical Learning, Second Edition*.

[2] Tom M. Mitchell, *Machine Learning: A multistrategy approach, 1997 Edition*.

[3] Adult.csv: URL - https://www.kaggle.com/datasets/qizarafzaal/adult-dataset