

## HW\_04\_NaiveBayes\_ChatGPT

Poorvi Raut -20009560

The "breast cancer dataset" in CANVAS was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The features in the dataset, described below, have been categorized from 1 to 10. Use the Naïve Bayes methodology to develop a classification model for the Diagnosis.

Important: make sure your categories are represented by the "factor" data type in **Python** and delete the rows with missing values. Use 30% test 70% training data

Features Domain

```
-- -----  
Sample code number id number  
F1. Clump Thickness 1 - 10  
F2. Uniformity of Cell Size 1 - 10  
F3. Uniformity of Cell Shape 1 - 10  
F4. Marginal Adhesion 1 - 10  
F5. Single Epithelial Cell Size 1 - 10  
F6. Bare Nuclei 1 - 10  
F7. Bland Chromatin 1 - 10  
F8. Normal Nucleoli 1 - 10  
F9. Mitoses 1 - 10  
Diagnosis Class: (2 for benign, 4 for malignant)
```

**Here below is the code generated of Implementing Naïve Bayes Algorithm using Python by CHATGPT**



To develop a Naïve Bayes classification model for the breast cancer dataset, we can follow the steps below:



1. Load the necessary libraries and the dataset.
2. Preprocess the dataset by deleting the rows with missing values and converting the categorical variables into the "factor" data type in Python.
3. Split the dataset into 30% test and 70% training data.
4. Train the Naïve Bayes classification model on the training data.
5. Test the model on the test data and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.

Here is the Python code to implement these steps:

```

import pandas as pd

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score

data = pd.read_csv("breast-cancer-wisconsin.csv")
print(data)

data = data.replace('?', np.nan)

data = data.dropna()
print(data)

X = data.drop(["Sample", "Class"], axis=1)
y = data["Class"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, r
andom_state=123)

print(X_train)
print(y_train)
print(X_test)
print(y_test)

clf = GaussianNB()
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
confusion_matrix_result = confusion_matrix(y_test, y_pred)
print(confusion_matrix_result)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

```

**After running the code on jupyter lab :**

**We get output as :**

698 897471 4 8 8 5 4 5 10 4 1 4

[683 rows x 11 columns]

	F1	F2	F3	F4	F5	F6	F7	F8	F9
696	5	10	10	3	7	3	8	10	2
257	3	1	1	1	2	1	2	1	1
489	6	3	2	1	3	4	4	1	1
95	1	1	1	1	2	1	3	1	1
692	3	1	1	1	2	1	1	1	1
..	..	..	..	..	..	..	..	..	..
100	10	3	5	1	10	5	3	10	2
336	6	5	5	8	4	10	3	4	1
396	3	1	1	1	2	1	3	1	1
379	5	3	4	1	4	1	3	1	1
525	3	1	1	2	2	1	1	1	1

[478 rows x 9 columns]

696	4
257	2
489	4
95	2
692	2

..	..
100	4
336	4
396	2
379	2
525	2

Name: Class, Length: 478, dtype: int64

	F1	F2	F3	F4	F5	F6	F7	F8	F9
154	1	1	1	1	2	1	1	1	1
687	3	1	1	1	2	1	2	3	1
80	2	2	2	1	1	1	7	1	1
269	1	1	1	1	2	1	3	1	1
136	4	1	1	1	2	1	2	1	1
..	..	..	..	..	..	..	..	..	..
285	8	10	10	10	8	10	10	7	3
289	5	6	6	8	6	10	4	10	4
177	5	10	10	3	8	1	5	10	3
180	1	1	1	1	1	1	3	1	1
215	8	7	8	7	5	5	5	10	2

[205 rows x 9 columns]

154	2
687	2
80	2
269	2
136	2

..	..
285	4

```
289      4
177      4
180      2
215      4
Name: Class, Length: 205, dtype: int64
[[129      3]
 [  0     73]]
Accuracy: 0.9853658536585366
```