

I. TUJUAN

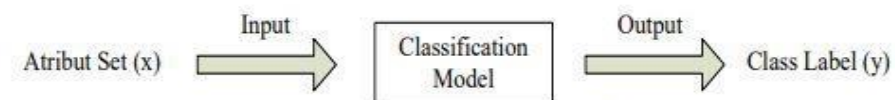
- Praktikan mampu memahami algoritma *naïve bayes*.
- Praktikan mampu mengimplementasikan algoritma *Naive Bayes*.

II. ALAT DAN BAHAN

- Laptop
- Pycharm*
- Python*
- Modul
- Microsoft Word*

III. TEORI DASAR

Naive Bayes merupakan sebuah pengklasifikasian probabilistik sederhana yang menghitung sekumpulan probabilitas dengan menjumlahkan frekuensi dan kombinasi nilai dari *dataset* yang diberikan. Algoritma menggunakan teorema *Bayes* dan mengasumsikan semua atribut independen atau tidak saling ketergantungan yang diberikan oleh nilai pada variabel kelas.



Gambar 1. Blok Diagram Model Klasifikasi

Definisi lain mengatakan *Naive Bayes* merupakan pengklasifikasian dengan metode probabilitas dan statistik yang dikemukakan oleh ilmuwan Inggris Thomas Bayes, yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya. *Naive Bayes* didasarkan pada asumsi penyederhanaan bahwa nilai atribut secara kondisional saling bebas jika diberikan nilai output. Dengan kata lain, diberikan nilai output, probabilitas mengamati secara bersama adalah produk dari probabilitas individu. Keuntungan penggunaan *Naive Bayes*

adalah bahwa metode ini hanya membutuhkan jumlah data pelatihan (*Training Data*) yang kecil untuk menentukan estimasi parameter yang diperlukan dalam proses pengklasifikasian. *Naive Bayes* sering bekerja jauh lebih baik dalam kebanyakan situasi dunia nyata yang kompleks dari pada yang diharapkan.

Persamaan dari teorema Bayes adalah

$$P(H|X) = \frac{P(X|H) \cdot P(H)}{P(X)} \quad (1)$$

Di mana :

X : Data dengan *class* yang belum diketahui

H : Hipotesis data merupakan suatu *class* spesifik

$P(H|X)$: Probabilitas hipotesis H berdasar kondisi X (posteriori probabilitas)

$P(H)$: Probabilitas hipotesis H (prior probabilitas)

$P(X|H)$: Probabilitas X berdasarkan kondisi pada hipotesis H

$P(X)$: Probabilitas X

Untuk menjelaskan metode *Naive Bayes*, perlu diketahui bahwa proses klasifikasi memerlukan sejumlah petunjuk untuk menentukan kelas apa yang cocok bagi sampel yang dianalisis tersebut. Karena itu, metode *Naive Bayes* di atas disesuaikan sebagai berikut:

$$P(C|F1 \dots Fn) = \frac{P(C)P(F1 \dots Fn|C)}{P(F1 \dots Fn)} \quad (2)$$

Di mana Variabel C merepresentasikan kelas, sementara variabel $F1 \dots Fn$ merepresentasikan karakteristik petunjuk yang dibutuhkan untuk melakukan klasifikasi. Maka rumus tersebut menjelaskan bahwa peluang masuknya sampel karakteristik tertentu dalam kelas C (Posterior) adalah peluang munculnya kelas C (sebelum masuknya sampel tersebut, seringkali disebut *prior*), dikali dengan peluang kemunculan karakteristik, karakteristik sampel pada kelas C (disebut juga *likelihood*), dibagi dengan peluang kemunculan karakteristik-karakteristik sampel secara global (disebut juga *evidence*). Karena itu, rumus di atas dapat pula ditulis secara sederhana sebagai berikut:

$$Posterior = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \quad (3)$$

Nilai *Evidence* selalu tetap untuk setiap kelas pada satu sampel. Nilai dari posterior tersebut nantinya akan dibandingkan dengan nilai-nilai posterior kelas lainnya untuk menentukan ke kelas apa suatu sampel akan diklasifikasikan. Penjabaran lebih lanjut rumus Bayes tersebut dilakukan dengan menjabarkan $(C|F_1, \dots, F_n)$ menggunakan aturan perkalian sebagai berikut:

$$\begin{aligned} P(C|F_1, \dots, F_n) &= P(C)P(F_1, \dots, F_n|C) \\ &= P(C)P(F_1|C)P(F_2, \dots, F_n|C, F_1) \\ &= P(C)P(F_1|C)P(F_2|C, F_1)P(F_3, \dots, F_n|C, F_1, F_2) \\ &= (C)P(F_1|C)P(F_2|C, F_1)P(F_3|C, F_1, F_2)P(F_4, \dots, F_n|C, F_1, F_2, F_3) \\ &= P(C)P(F_1|C)P(F_2|C, F_1)P(F_3|C, F_1, F_2) \dots P(F_n|C, F_1, F_2, F_3, \dots, F_{n-1}) \quad (4) \end{aligned}$$

Dapat dilihat bahwa hasil penjabaran tersebut menyebabkan semakin banyak dan semakin kompleksnya faktor - faktor syarat yang mempengaruhi nilai probabilitas, yang hampir mustahil untuk dianalisa satu persatu. Akibatnya, perhitungan tersebut menjadi sulit untuk dilakukan. Di sinilah digunakan asumsi independensi yang sangat tinggi (naif), bahwa masing-masing petunjuk ($F_1, F_2 \dots F_n$) saling bebas (independen) satu sama lain. Dengan asumsi tersebut, maka berlaku suatu kesamaan sebagai berikut:

$$P(F_i|F_j) = \frac{P(F_i \cap F_j)}{P(F_j)} = \frac{P(F_i)P(F_j)}{P(F_j)} = P(F_i) \quad (5)$$

Untuk $i \neq j$, sehingga

$$P(F_i|C, F_j) = P(F_i|C) \quad (6)$$

Persamaan di atas merupakan model dari teorema *Naive Bayes* yang selanjutnya akan digunakan dalam proses klasifikasi. Untuk klasifikasi dengan data kontinu digunakan rumus Densitas Gauss :

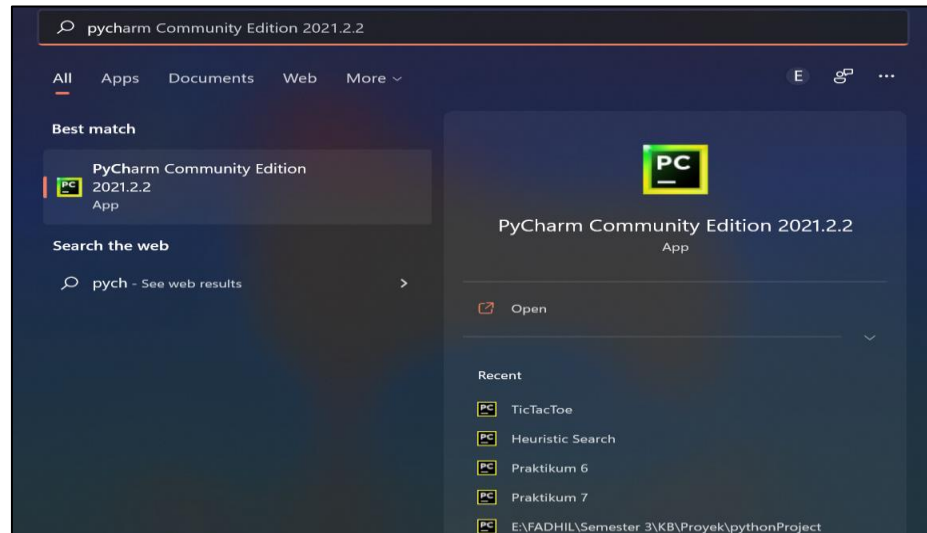
$$P(X_i = x_i | Y = y_j) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} e^{-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}} \quad (7)$$

Di mana :

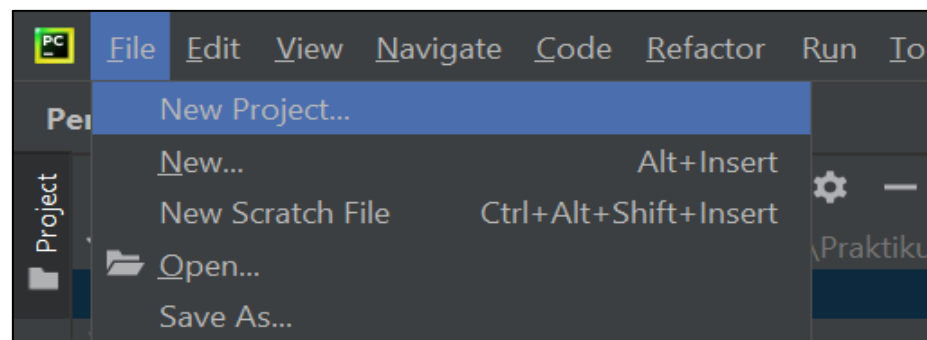
- P : Peluang
- X_i : Atribut ke i
- x_i : Nilai atribut ke i
- Y : Kelas yang dicari
- y_i : Sub kelas Y yang dicari
- μ : *mean*, menyatakan rata – rata dari seluruh atribut
- σ : *Deviasi standar*, menyatakan varian dari seluruh atribut.

IV. LANGKAH KERJA

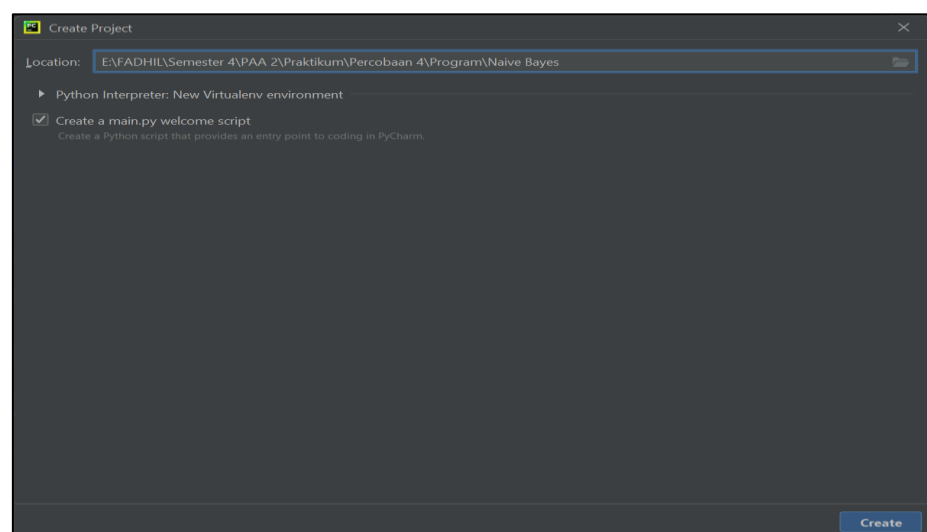
- a. Buka aplikasi *pycharm*.



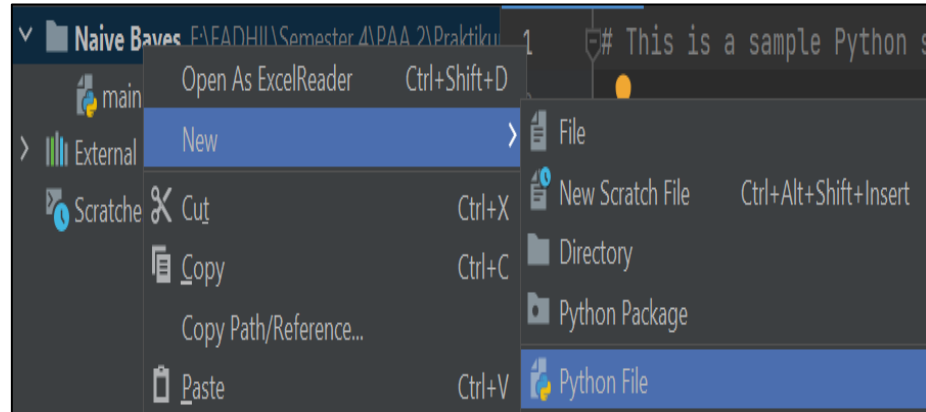
- b. Klik *file, new project*.



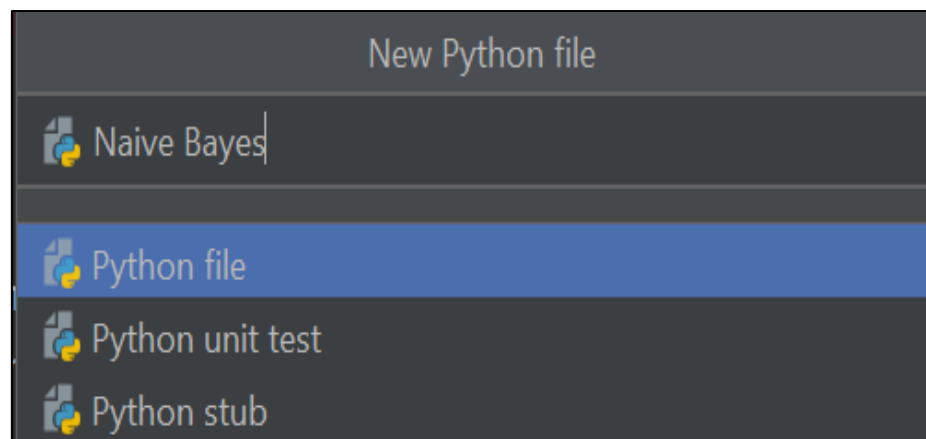
- c. Tentukanlah letak *file project* dan beri nama *project* lalu klik *create*.



- d. Klik kanan pada nama *file* yang telah di buat, kemudian pilih *new file*.



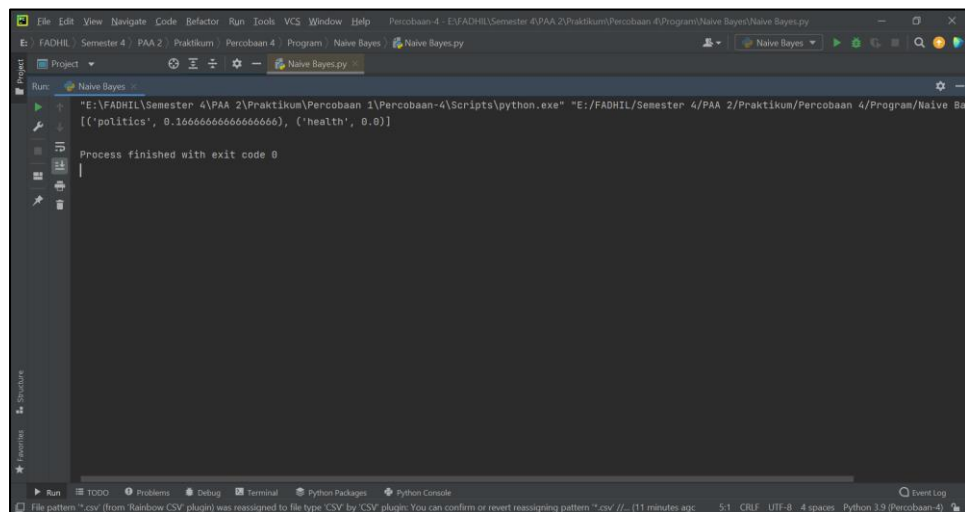
- e. Tentukan nama *file project* yang di buat.



- f. Masukkan kode program berikut.

```
1 from naiveBayesClassifier import tokenizer
2 from naiveBayesClassifier.trainer import Trainer
3 from naiveBayesClassifier.classifier import Classifier
4
5 newsTrainer = Trainer(tokenizer.Tokenizer(stop_words=[], signs_to_remove=["?!#%&"]))
6
7 newsSet = [
8     {'text': 'not to eat too much is not enough to lose weight', 'category': 'health'},
9     {'text': 'Russia is trying to invade Ukraine', 'category': 'politics'},
10    {'text': 'do not neglect exercise', 'category': 'health'},
11    {'text': 'Syria is the main issue, Fadhl says', 'category': 'politics'},
12    {'text': 'eat to lose weight', 'category': 'health'},
13    {'text': 'you should not eat much', 'category': 'health'}
14 ]
15 for news in newsSet:
16     newsTrainer.train(news['text'], news['category'])
17
18 newsClassifier = Classifier(newsTrainer.data, tokenizer.Tokenizer(stop_words=[], signs_to_remove=["?!#%&"]))
19
20 unknownInstance = "Fadhl"
21 classification = newsClassifier.classify(unknownInstance)
22
23 print(classification)
```

V. HASIL PERCOBAAN



```
Run: Naive Bayes
"E:\FADHIL\Semester 4\PAA 2\Praktikum\Percobaan 1\Percobaan-4\Scripts\python.exe" "E:\FADHIL\Semester 4\PAA 2\Praktikum\Percobaan 4\Program\Naive Bayes.py"
[['politics', 0.16666666666666666], ('health', 0.0)]
Process finished with exit code 0
```

VI. ANALISIS

Pada percobaan diatas dapat dianalisis bahwa untuk mengimplementasi algoritma *naive bayes*, pertama kita mengimport *library naiveBayesClassifier*, yang berfungsi memprediksi probabilitas pada masa depan berdasarkan pengalaman pada masa sebelumnya, Pada *library* tersebut kita mengimport lagi fungsi *classifier* dan *trainer*. masukkan kode program untuk fungsi *trainer* “*newsTrainer = Trainer(tokenizer.Tokenizer(stop_words = [], signs_to_remove = ["?!#%&"])),* pada *trainer* terdapat variabel(*Stop_words = [], signs_to_remove = ["?!#%&"]*) yang dimana akan dilakukan penyaringan data sebelum atau sesudah pemrosesan bahasa alami(teks). Terdapat juga variabel “*newsSet =[{‘text’: ‘Syria is the main issue, Fadhil says’, ‘category’: ‘politics’}]*”, sebagai data pelatihan yang terdiri atas *text* dan *category*.

Variabel “*for news in newsSet:*” untuk melakukan perulangan pada *newsSet*, “*newsTrainer.train[‘text’], news[‘category’])*” merupakan kondisi untuk memenuhi perulangan yang dilakukan pada *newsSet*. “*newsClassifier =*

