Naïve Bayes

```
In [1]: import pandas as pd
```

```
In [2]: dataset = pd.read_csv('heart_disease.csv')
```

```
In [3]: dataset
```

Out[3]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up	0

918 rows × 12 columns

Kode ini mengimpor pustaka pandas dan membaca dataset dari file CSV bernama "heart_disease.csv". Setelah dataset dibaca, hasilnya disimpan dalam variabel dataset.

```
In [4]: dataset.info
```

Out[4]:

```
<bound method DataFrame.info of      Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG  \
0     40   M           ATA        140          289          0     Normal
1     49   F           NAP        160          180          0     Normal
2     37   M           ATA        130          283          0         ST
3     48   F           ASY        138          214          0     Normal
4     54   M           NAP        150          195          0     Normal
..   ...  ..           ...        ...          ...        ...        ...
913   45   M            TA        110          264          0     Normal
914   68   M           ASY        144          193          1     Normal
915   57   M           ASY        130          131          0     Normal
916   57   F           ATA        130          236          0        LVH
917   38   M           NAP        138          175          0     Normal

     MaxHR ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0      172              N      0.0       Up             0
1      156              N      1.0     Flat             1
2       98              N      0.0       Up             0
3      108              Y      1.5     Flat             1
4      122              N      0.0       Up             0
..     ...            ...      ...      ...           ...
913    132              N      1.2     Flat             1
914    141              N      3.4     Flat             1
915    115              Y      1.2     Flat             1
916    174              N      0.0     Flat             1
917    173              N      0.0       Up             0

[918 rows x 12 columns]>
```

dataset.info(), yang menginformasikan tipe data, jumlah entri, dan nilai non-null.

```
In [5]: dataset.Sex.unique()
Out[5]: array(['M', 'F'], dtype=object)

In [6]: dataset.ChestPainType.unique()
Out[6]: array(['ATA', 'NAP', 'ASY', 'TA'], dtype=object)

In [7]: dataset.RestingECG.unique()
Out[7]: array(['Normal', 'ST', 'LVH'], dtype=object)

In [8]: dataset.ExerciseAngina.unique()
Out[8]: array(['N', 'Y'], dtype=object)

In [9]: dataset.ST_Slope.unique()
Out[9]: array(['Up', 'Flat', 'Down'], dtype=object)
```

Kode ini menampilkan nilai khusus untuk kolom-kolom tertentu dalam dataset tertentu; contohnya, jenis kelamin (Sex), tipe nyeri dada (ChestPainType), hasil elektrokardiogram istirahat (RestingECG), dan lainnya.

```
In [10]: from sklearn.preprocessing import LabelEncoder

In [11]: for col in ['Sex','ChestPainType','RestingECG','ExerciseAngina','ST_Slope']:
             label_encoder = LabelEncoder()
             dataset[col] = label_encoder.fit_transform(dataset[col])
```

```
In [12]: dataset
```

Out[12]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	1	1	140	289	0	1	172	0	0.0	2	0
1	49	0	2	160	180	0	1	156	0	1.0	1	1
2	37	1	1	130	283	0	2	98	0	0.0	2	0
3	48	0	0	138	214	0	1	108	1	1.5	1	1
4	54	1	2	150	195	0	1	122	0	0.0	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	1	3	110	264	0	1	132	0	1.2	1	1
914	68	1	0	144	193	1	1	141	0	3.4	1	1
915	57	1	0	130	131	0	1	115	1	1.2	1	1
916	57	0	1	130	236	0	0	174	0	0.0	1	1
917	38	1	2	138	175	0	1	173	0	0.0	2	0

918 rows × 12 columns

Nilai kategori dalam kolom tertentu diubah menjadi bilangan bulat dengan menggunakan LabelEncoder dari scikit-learn.

```
In [13]: x = dataset.drop('HeartDisease', axis = 1)
         y = dataset['HeartDisease']

In [14]: x
```

Out[14]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	40	1	1	140	289	0	1	172	0	0.0	2
1	49	0	2	160	180	0	1	156	0	1.0	1
2	37	1	1	130	283	0	2	98	0	0.0	2
3	48	0	0	138	214	0	1	108	1	1.5	1
4	54	1	2	150	195	0	1	122	0	0.0	2
...	...	...	...	...	...	...	...	...	...	...	...
913	45	1	3	110	264	0	1	132	0	1.2	1
914	68	1	0	144	193	1	1	141	0	3.4	1
915	57	1	0	130	131	0	1	115	1	1.2	1
916	57	0	1	130	236	0	0	174	0	0.0	1
917	38	1	2	138	175	0	1	173	0	0.0	2

918 rows × 11 columns

```
In [15]: y

Out[15]: 0      0
         1      1
         2      0
         3      1
         4      0
               ..
         913    1
         914    1
         915    1
         916    1
         917    0
         Name: HeartDisease, Length: 918, dtype: int64
```

Kode ini memisahkan variabel independen (x) dan variabel dependen (y) dari kumpulan data. Variabel independen (x) mengandung semua kolom kecuali kolom "Heart disease", sedangkan variabel dependen (y) mengandung hanya kolom "Heart Disease".

```
In [16]: from sklearn.model_selection import train_test_split

In [17]: x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.15)

In [18]: x_train.shape

Out[18]: (780, 11)

In [19]: y_train.shape

Out[19]: (780,)

In [20]: x_test.shape

Out[20]: (138, 11)

In [21]: y_test.shape

Out[21]: (138,)
```

Dengan menggunakan kode train_test_split, dataset dibagi menjadi data latihan (x_train dan y_train) dan data uji (x_test dan y_test).

```
In [22]: from sklearn.naive_bayes import GaussianNB
         modelNB = GaussianNB()
         modelNB.fit(x_train, y_train)

Out[22]: GaussianNB()
```

Kode ini membuat objek model Naive Bayes dengan mengimport kelas GaussianNB dari scikit-learn dan melatihnya menggunakan data latih.

```
In [23]: hasil_prediksi = modelNB.predict(x_test)

In [24]: from sklearn.metrics import accuracy_score
         print('Akurasi Model: ', accuracy_score(y_test, hasil_prediksi))

         Akurasi Model:  0.8695652173913043
```

kode ini menggunakan model yang telah dilatih untuk melakukan prediksi pada data uji, dan kemudian menggunakan fungsi accuracy_score dari scikit-learn untuk menghitung dan mencetak akurasi model.

Regresi

```
In [1]: import pandas as pd
```

```
In [2]: data1 = pd.read_csv('Students_performance.csv')
        data1.head()
```

Out[2]:

	Study Hours (X)	Exam Scores (Y)
0	1.5	60
1	2.0	65
2	2.5	73
3	3.0	75
4	2.0	62

```
In [3]: data1
```

Out[3]:

	Study Hours (X)	Exam Scores (Y)
0	1.5	60
1	2.0	65
2	2.5	73
3	3.0	75
4	2.0	62
5	3.5	85
6	4.0	92
7	3.0	78
8	1.8	63
9	4.5	95
10	2.7	70
11	3.2	80
12	2.1	66
13	4.0	88
14	2.9	75
15	1.6	58
16	3.8	89
17	2.3	69
18	4.2	92
19	3.5	82
20	3.0	76
21	2.5	67
22	3.7	87
23	2.8	72
24	4.1	90
25	3.3	80
26	2.4	68
27	3.6	84
28	2.2	70
29	1.9	61
30	3.9	91
31	2.6	74
32	4.3	93
33	3.4	79
34	2.7	71
35	1.7	59
36	4.4	94
37	3.2	77
38	2.0	61
39	3.1	84
40	2.3	66
41	4.0	89
42	3.7	73
43	2.5	63
44	4.1	95
45	3.0	78

Kode ini menggunakan Pandas untuk membaca dataset dari file CSV ('Students_performance.csv'), menampilkan lima baris pertama dari dataset.

```
In [4]: data1.corr()
```

Out[4]:

	Study Hours (X)	Exam Scores (Y)
Study Hours (X)	1.000000	0.943875
Exam Scores (Y)	0.943875	1.000000

menghitung korelasi antara variabel 'Study Hours (X)' dan 'Exam Scores (Y).

```
import numpy as np

study_hours = data1['Study Hours (X)'].values[:, np.newaxis]
examscores = data1['Exam Scores (Y)'].values
```

```
In [6]: print(study_hours)
```

```
[[1.5]
 [2. ]
 [2.5]
 [3. ]
 [2. ]
 [3.5]
 [4. ]
 [3. ]
 [1.8]
 [4.5]
 [2.7]
 [3.2]
 [2.1]
 [4. ]
 [2.9]
 [1.6]
 [3.8]
 [2.3]
 [4.2]
 [3.5]
 [3. ]
 [2.5]
 [3.7]
 [2.8]
 [4.1]
 [3.3]
 [2.4]
 [3.6]
 [2.2]
 [1.9]
 [3.9]
 [2.6]
 [4.3]
 [3.4]
```

```
[2.7]
[1.7]
[4.4]
[3.2]
[2. ]
[3.1]
[2.3]
[4. ]
[3.7]
[2.5]
[4.1]
[3. ]
[1.8]
[4.5]
[3.3]
[2.2]]
```

kode ini memisahkan variabel 'Study Hours (X)' dan 'Exam Scores (Y)' untuk mempersiapkan data untuk regresi linear.

```
from sklearn.linear_model import LinearRegression

model1 = LinearRegression()
model1.fit(study_hours, examscores)
```

```
LinearRegression()
```

Kode ini menggunakan LinearRegression dari Scikit-learn untuk membuat objek model regresi linear dan melatihnya menggunakan data yang telah disiapkan.

```
In [8]: study_hours_test = [[3],[4.5]]
        pred_exam = model1.predict(study_hours_test)
        print("Prediksi Score: ", pred_exam)
```

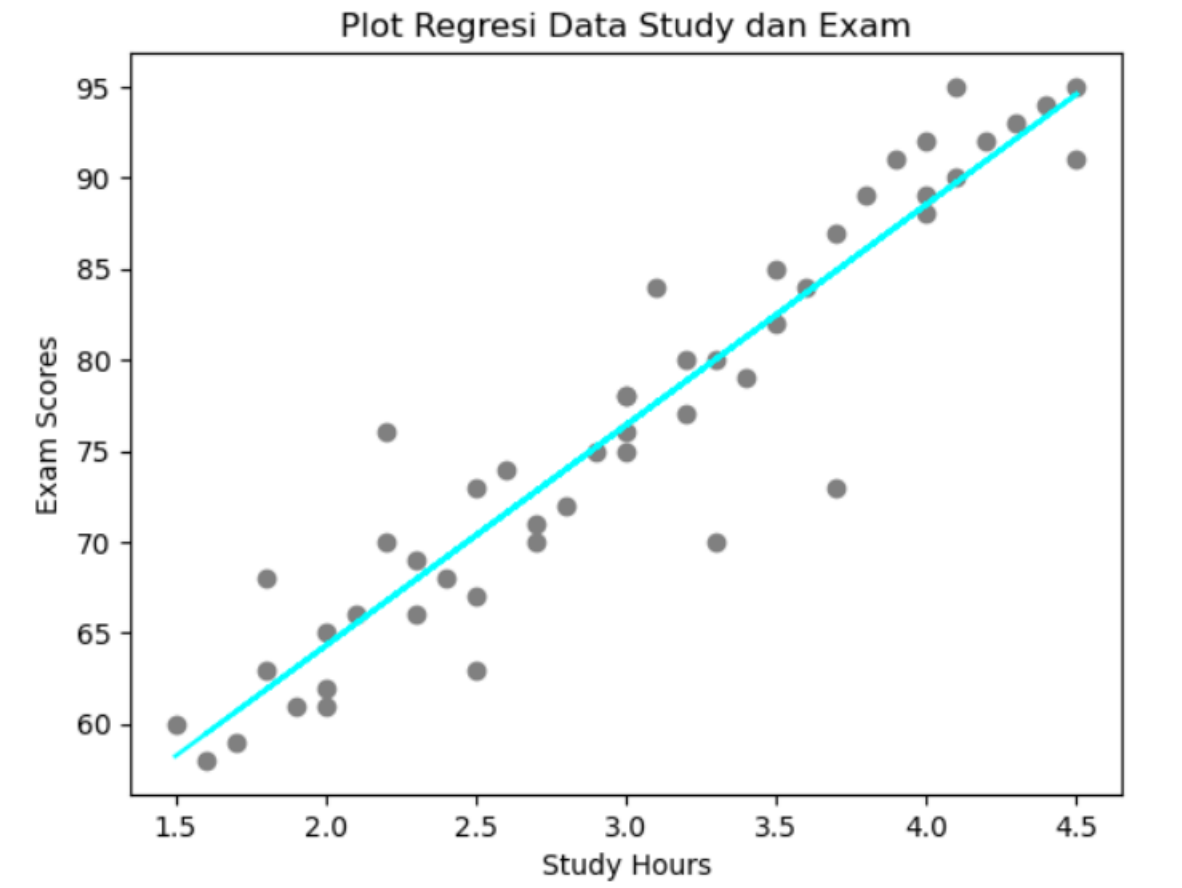
```
Prediksi Score:  [76.40727289 94.58905099]
```

Kode ini menggunakan model yang telah dilatih untuk melakukan prediksi skor ujian berdasarkan jam belajar (3 dan 4.5 jam).

```
In [9]: print("Koefisien: ", model1.coef_)
        print("Intercept: ", model1.intercept_)

        Koefisien:  [12.1211854]
        Intercept:  40.04371669027397
```

Kode ini mencetak koefisien dan intercept dari model regresi linear.

```
In [14]: import matplotlib.pyplot as plt

         # Assuming 'Study Hours (X)' exists in data1
         study_hours = data1['Study Hours (X)'].values.reshape(-1, 1)
         prediksi_score = model1.predict(study_hours)

         dataframe_baru = pd.DataFrame({'Study Hours (X)': data1['Study Hours (X)'], 'Exam Scores (Y)': prediksi_score})
         plt.scatter(data1['Study Hours (X)'], data1['Exam Scores (Y)'], color='grey')
         plt.plot(dataframe_baru['Study Hours (X)'], dataframe_baru['Exam Scores (Y)'], color='cyan')
         plt.xlabel('Study Hours')
         plt.ylabel('Exam Scores')
         plt.title('Plot Regresi Data Study dan Exam')
         plt.show()
```



Kode ini menggunakan matplotlib untuk membuat scatter plot dari data aktual dan garis regresi linear yang diprediksi oleh model. Scatter plot menunjukkan distribusi data aktual, sedangkan garis regresi menunjukkan hubungan prediksi model antara jam belajar dan skor ujian.