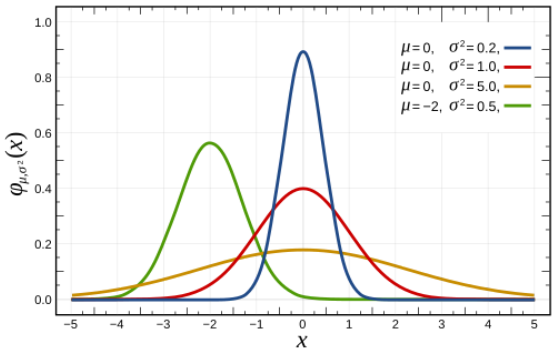
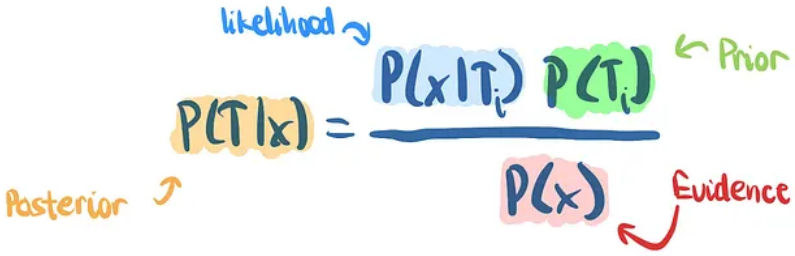
# Gaussian Naive Bayes (GNB)

## Cara Kerja

Model Gaussian Naive Bayes bekerja dengan prinsip distribusi normal (gaussian) dan teorema bayes.




Distribusi Normal/Gaussian
Sumber: https://en.wikipedia.org/wiki/Normal_distribution




Teorema Bayes
Sumber:
https://medium.com/@kashishdafe0410/gaussian-naive-bayes-understanding-the-basics-and-applications-52098087b963

Pada tahap fitting, model ini menghitung mean, variansi, dan prior probability dari setiap fitur dalam dataset. Mean dan variansi mendeskripsikan distribusi gaussian dari setiap fitur. Sedangkan, prior probability adalah probabilitas kemunculan sebuah kelas dalam dataset yang dinyatakan sebagai jumlah kemunculan kelas tersebut dibagi dengan jumlah data.

$$P(class|features) = \frac{P(class) \times P(features|class)}{P(features)}$$

Sumber:
https://medium.com/@kashishdafe0410/gaussian-naive-bayes-understanding-the-basics-and-applications-52098087b963

Pada tahap prediksi, pada dasarnya model ini menghitung posterior probability dari setiap kelas apabila diberikan tuple fitur yang ingin diprediksi tersebut lalu kelas target ditentukan sebagai kelas dengan posterior probability tertinggi. Karena P(features) bersifat konstan, maka kalkulasi suku ini dapat diabaikan dalam perbandingan yang dilakukan model ini. Sehingga, kita bisa menggunakan proporsionalitas berikut:

$$P(class \mid features) \propto P(class) \times P(features \mid class)$$
$$score(class) = P(class) \times P(features \mid class)$$

Dengan asumsi naive yakni setiap fitur bersifat saling lepas, maka

$$P(features \mid class) = P(feature1 \mid class) \times P(feature2 \mid Class) \times \ldots$$

Namun, perkalian berulang ini dapat menghasilkan nilai yang sangat kecil dan tidak aman bagi komputer (rentan terhadap underflow). Dengan demikian, digunakan logaritma yang dapat mencegah bilangan hasil kalkulasi yang terlalu kecil dan tetap mempertahankan deskripsi proporsi probabilitas setiap kelas.

$$\log(score(class)) = \log(P(class)) + \log(P(feature1 \mid class)) \times \log(P(feature2 \mid Class)) + \ldots$$

# Evaluasi Model

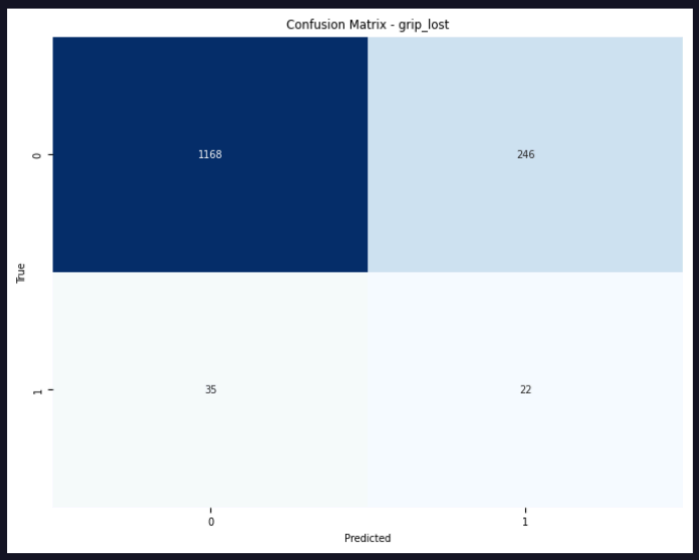
| Evaluasi | Hasil |
|---|---|
| **Model from Scratch** | |

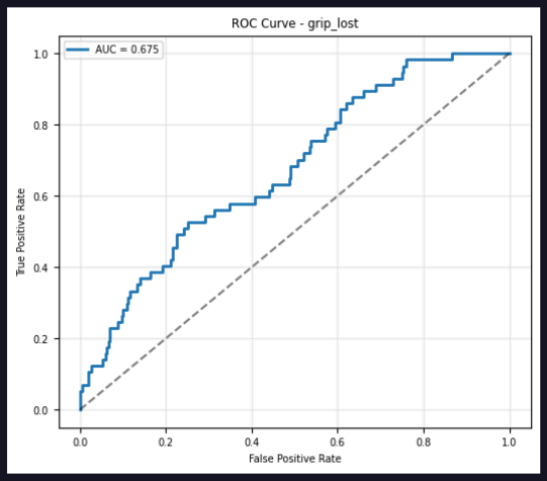
Hold-out validation untuk grip_lost

```
============================================================
Hold-Out Validation for grip_lost:
============================================================
Training time: 0.008644342422485352 seconds
Prediction time: 0.0015087127685546875 seconds
              precision    recall  f1-score   support

           0       0.97      0.83      0.89      1414
           1       0.08      0.39      0.14        57

    accuracy                           0.81      1471
   macro avg       0.53      0.61      0.51      1471
weighted avg       0.94      0.81      0.86      1471
```



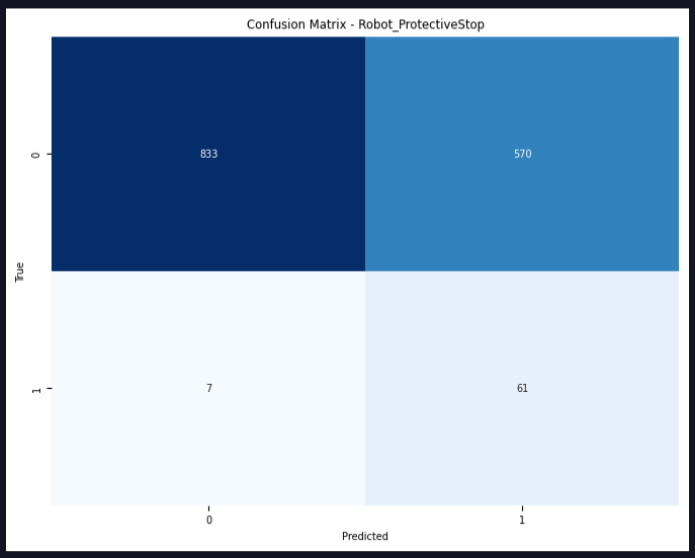

```
AUC-ROC: 0.6750
```

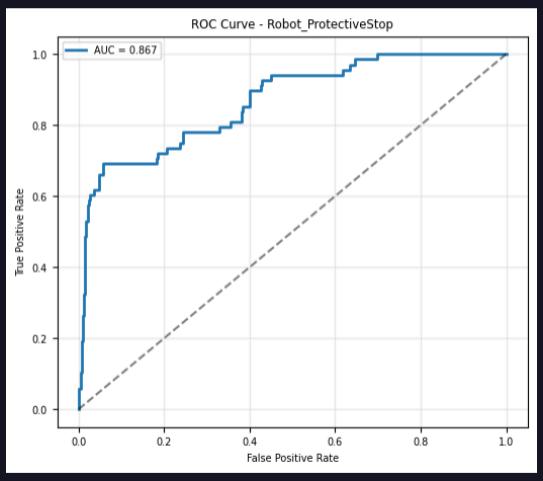
Hold out validation untuk Robot_ProtectiveStop

```
============================================================
Hold-Out Validation for Robot_ProtectiveStop:
============================================================
Training time: 0.004000663757324219 seconds
Prediction time: 0.0005049705505371094 seconds
              precision    recall  f1-score   support

           0       0.99      0.59      0.74      1403
           1       0.10      0.90      0.17        68

    accuracy                           0.61      1471
   macro avg       0.54      0.75      0.46      1471
weighted avg       0.95      0.61      0.72      1471
```




```
AUC-ROC: 0.8673
```

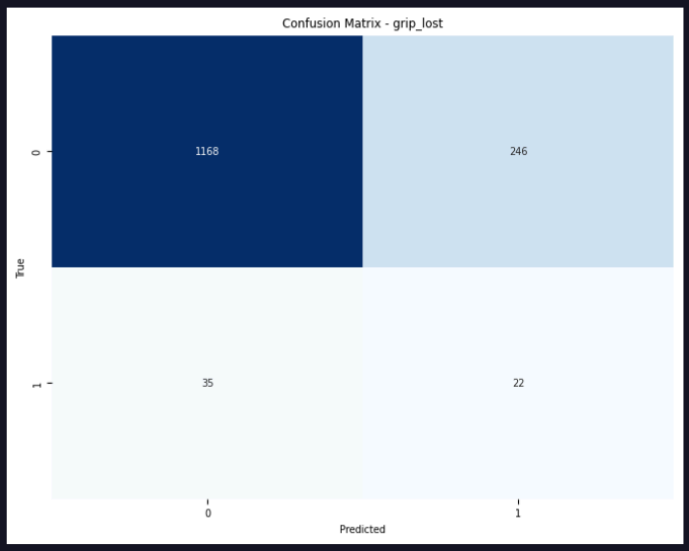
| | |
|---|---|
| K-fold cross validation untuk grip_lost | ```<br>============================================================<br>Cross Validation for grip_lost:<br>============================================================<br>Model Performance (5-Fold Cross Validation):<br><br>fit_time: [0.00399828 0.00400329 0.0039959  0.00465727 0.00299287]<br><br>score_time: [0.00676703 0.00705862 0.00553346 0.0040307  0.00551891]<br><br>test_precision: [0.08208955 0.10408922 0.06521739 0.08064516 0.07727273]<br>Average test_precision: 0.08<br><br>test_recall: [0.38596491 0.5        0.34090909 0.47619048 0.38636364]<br>Average test_recall: 0.42<br><br>test_f1: [0.13538462 0.17230769 0.10948905 0.13793103 0.12878788]<br>Average test_f1: 0.14<br>``` |
| K-fold cross validation untuk Robot_ProtectiveStop | ```<br>============================================================<br>Cross Validation for Robot_ProtectiveStop:<br>============================================================<br>Model Performance (5-Fold Cross Validation):<br><br>fit_time: [0.00551987 0.00351691 0.00301313 0.00300336 0.0030005 ]<br><br>score_time: [0.00500727 0.00453973 0.00532722 0.00552273 0.00508595]<br><br>test_precision: [0.09667195 0.06942393 0.08580343 0.06435644 0.06766917]<br>Average test_precision: 0.08<br><br>test_recall: [0.89705882 0.88679245 0.91666667 0.82978723 0.9       ]<br>Average test_recall: 0.89<br><br>test_f1: [0.17453505 0.12876712 0.15691869 0.1194487  0.12587413]<br>Average test_f1: 0.14<br>``` |
| **Model Scikit-Learn** | |

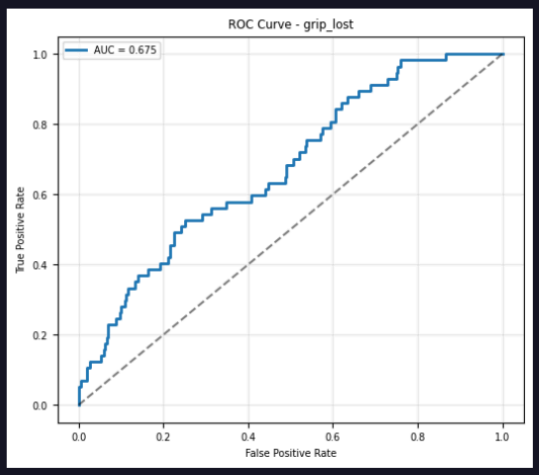
Hold-out validation untuk grip_lost

```
============================================================
Hold-Out Validation for grip_lost:
============================================================
Training time: 0.00756382942199707 seconds
Prediction time: 0.0010018348693847656 seconds
              precision    recall  f1-score   support

           0       0.97      0.83      0.89      1414
           1       0.08      0.39      0.14        57

    accuracy                           0.81      1471
   macro avg       0.53      0.61      0.51      1471
weighted avg       0.94      0.81      0.86      1471
```



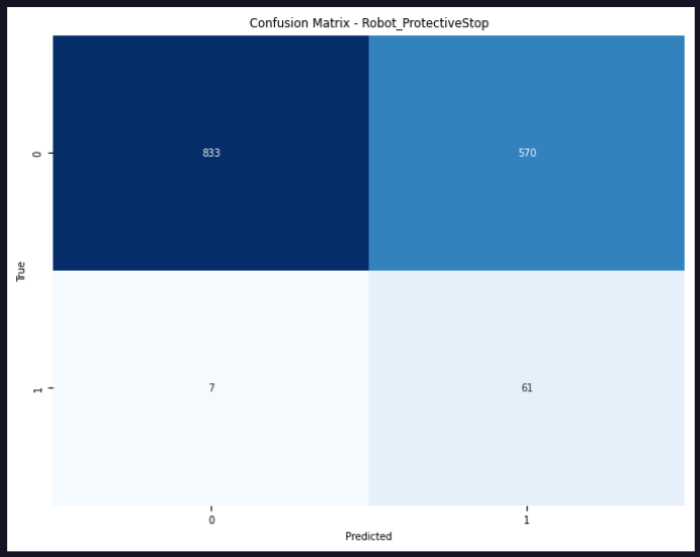

```
AUC-ROC: 0.6750
```

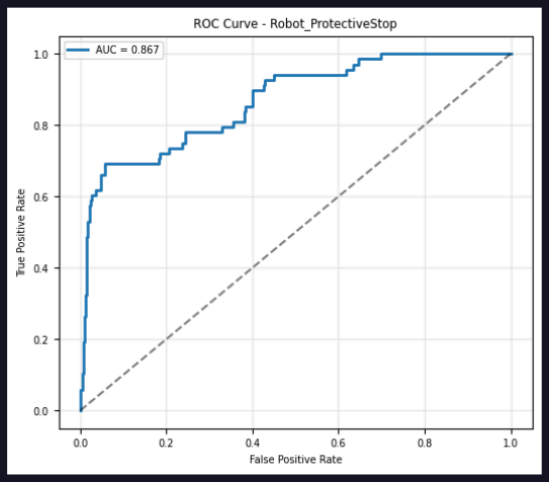
Hold out validation untuk Robot_ProtectiveStop

```
============================================================
Hold-Out Validation for Robot_ProtectiveStop:
============================================================
Training time: 0.005038261413574219 seconds
Prediction time: 0.0010075569152832031 seconds
              precision    recall  f1-score   support

           0       0.99      0.59      0.74      1403
           1       0.10      0.90      0.17        68

    accuracy                           0.61      1471
   macro avg       0.54      0.75      0.46      1471
weighted avg       0.95      0.61      0.72      1471
```




```
AUC-ROC: 0.8673
```

| | |
|---|---|
| K-fold cross validation untuk grip_lost | ```<br>============================================================<br>Cross Validation for grip_lost:<br>============================================================<br>Model Performance (5-Fold Cross Validation):<br><br>fit_time: [0.00800848 0.00550675 0.00551105 0.0070231  0.00357938]<br><br>score_time: [0.00508261 0.00600863 0.00599957 0.00551295 0.00460935]<br><br>test_precision: [0.08208955 0.10408922 0.06521739 0.08064516 0.07727273]<br>Average test_precision: 0.08<br><br>test_recall: [0.38596491 0.5        0.34090909 0.47619048 0.38636364]<br>Average test_recall: 0.42<br><br>test_f1: [0.13538462 0.17230769 0.10948905 0.13793103 0.12878788]<br>Average test_f1: 0.14<br>``` |
| K-fold cross validation untuk Robot_ProtectiveStop | ```<br>============================================================<br>Cross Validation for Robot_ProtectiveStop:<br>============================================================<br>Model Performance (5-Fold Cross Validation):<br><br>fit_time: [0.00599909 0.00499797 0.00653124 0.006073   0.00651765]<br><br>score_time: [0.00504899 0.00552297 0.00500035 0.00446248 0.00400472]<br><br>test_precision: [0.09667195 0.06942393 0.08580343 0.06435644 0.06766917]<br>Average test_precision: 0.08<br><br>test_recall: [0.89705882 0.88679245 0.91666667 0.82978723 0.9       ]<br>Average test_recall: 0.89<br><br>test_f1: [0.17453505 0.12876712 0.15691869 0.1194487  0.12587413]<br>Average test_f1: 0.14<br>``` |

Implementasi *from scratch* dan scikit-learn dari model GNB memiliki hasil metrik yang sama persis. Ini disebabkan karena algoritmanya yang *straight forward* dan tidak memiliki banyak parameter sehingga relatif mudah ditiru.

# Improvement

Nilai recall model GNB sudah cukup bagus, hanya saja precision-nya sangat rendah. Hal yang dapat dicoba adalah menerapkan prior probability awal yang dapat memberikan bobot yang berbeda terhadap setiap kelas. Metode preprocessing lain seperti oversampling yang umum digunakan dalam kasus unbalanced dataset juga dapat digunakan.