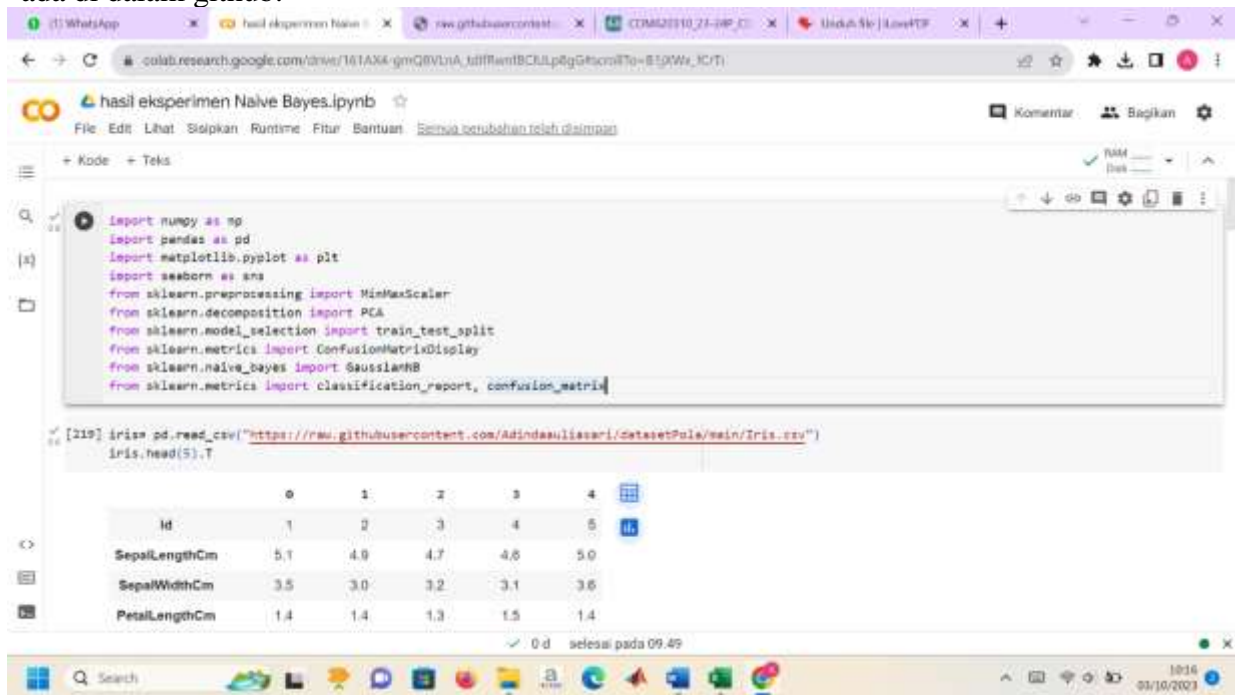


Nama : Adinda Aulia Sari

NPM : 2117051018

Kelas : D

Langkah Awal saya mengimport terlebih dahulu setelah itu memasukkan dataset iris yang ada di dalam github.



The screenshot shows a Jupyter Notebook titled "hasil eksperimen Naive Bayes.ipynb". The code cell contains the following imports and data loading:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix
```

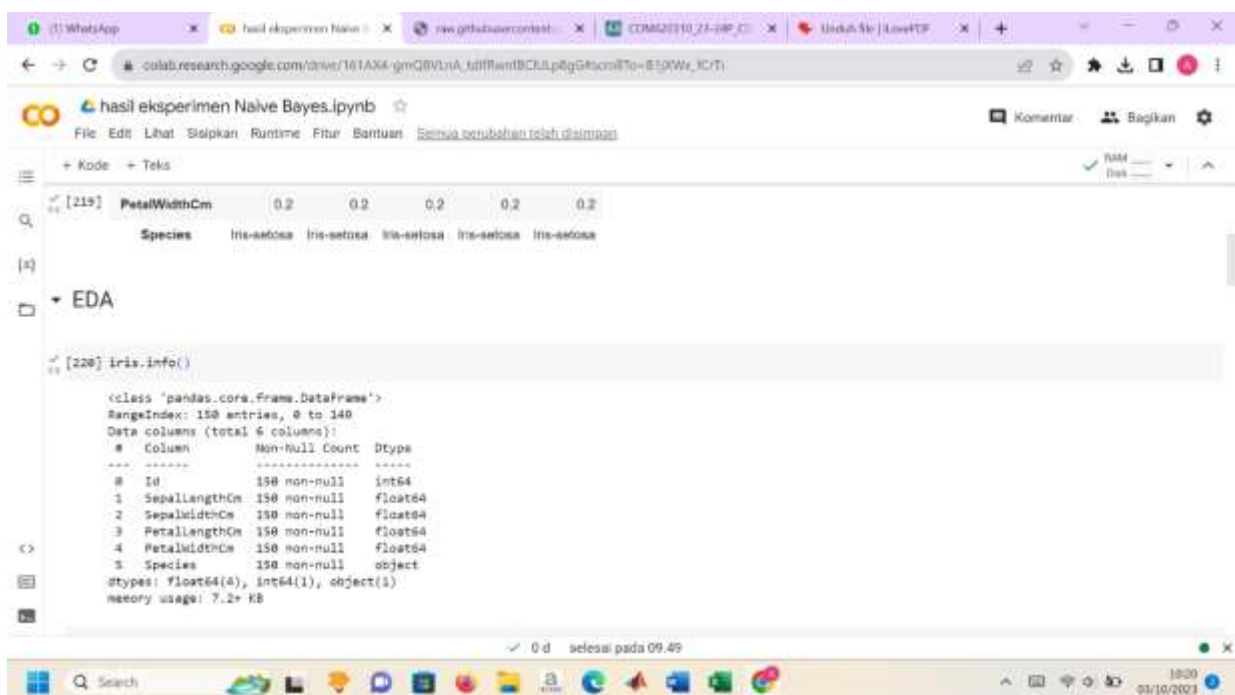
The next cell shows the dataset being loaded from a GitHub URL:

```
iris = pd.read_csv("https://raw.githubusercontent.com/AdindaAuliaSari/datasetPola/main/Iris.csv")
iris.head(5)
```

The output displays the first five rows of the dataset:

	0	1	2	3	4
Id	1	2	3	4	5
SepalLengthCm	5.1	4.9	4.7	4.6	5.0
SepalWidthCm	3.5	3.0	3.2	3.1	3.6
PetalLengthCm	1.4	1.4	1.3	1.5	1.4

Setelah itu menampilkan data set dengan mengetikkan fungsi “.info”



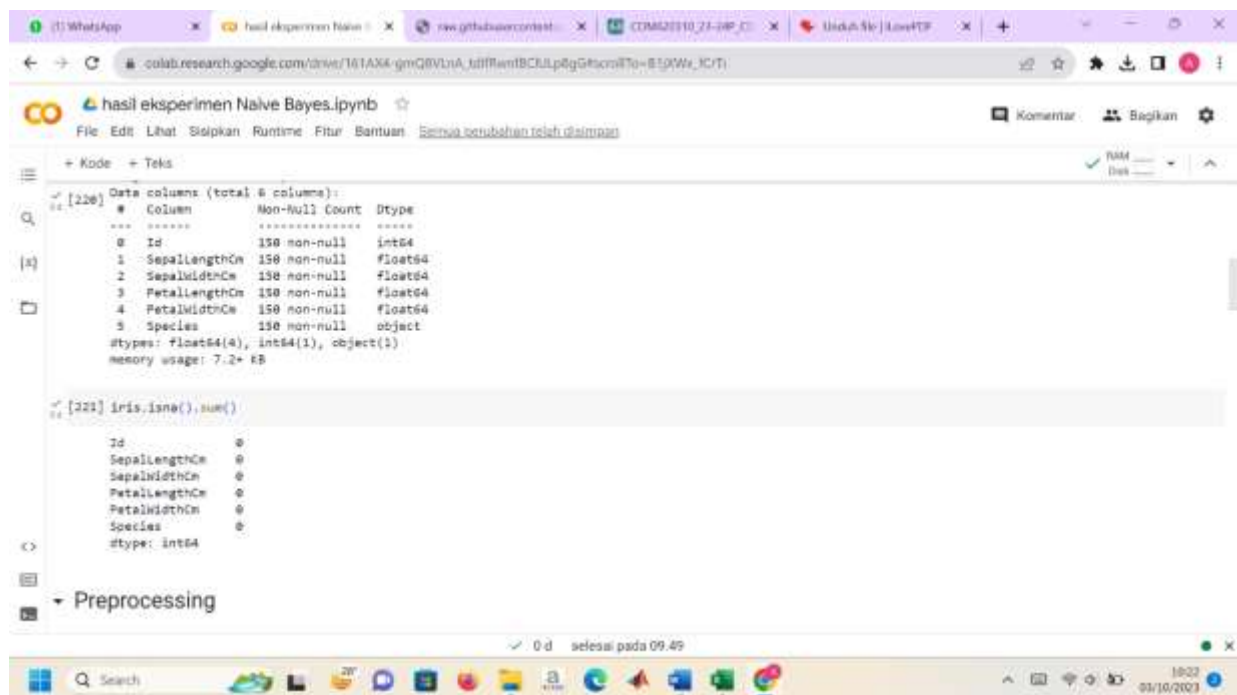
The screenshot shows the same Jupyter Notebook with the following code cell:

```
iris.info()
```

The output provides summary statistics for the dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  --
 0   Id              150 non-null   int64   
 1   SepalLengthCm   150 non-null   float64  
 2   SepalWidthCm    150 non-null   float64  
 3   PetalLengthCm   150 non-null   float64  
 4   PetalWidthCm    150 non-null   float64  
 5   Species         150 non-null   object  
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Selanjutnya saya akan mengecek nilai null dan deret yang kosong dengan fungsi `.isna`

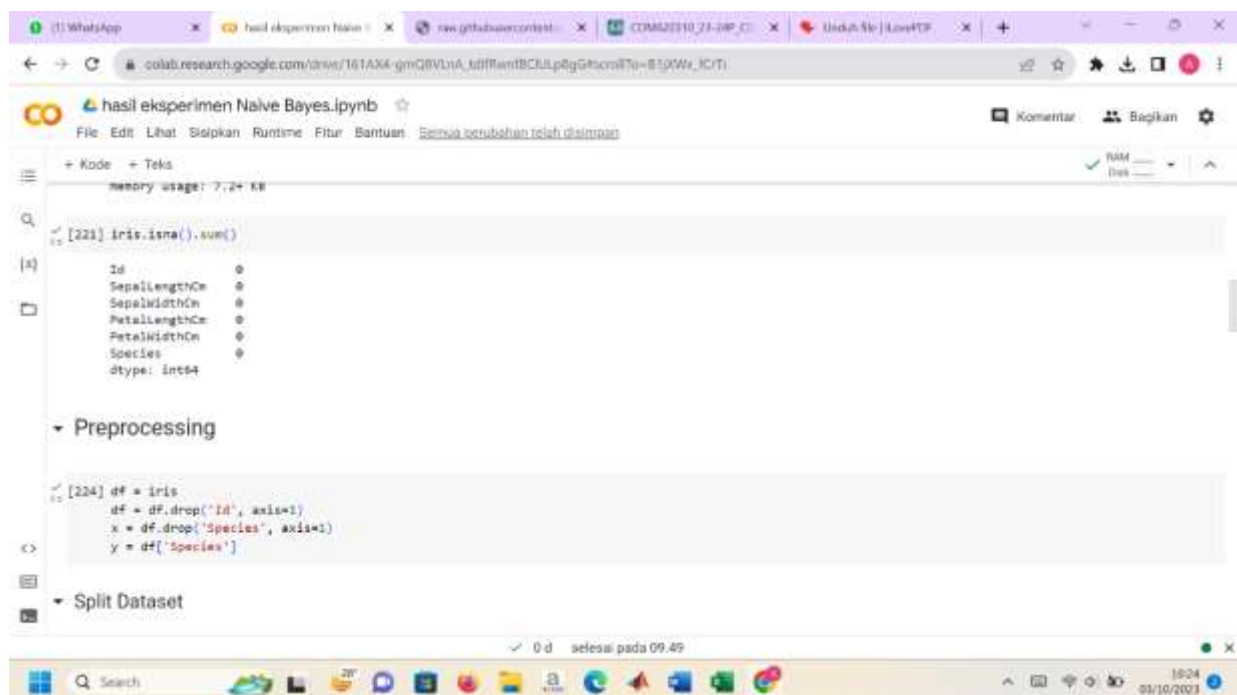


The screenshot shows a Jupyter Notebook titled "hasil eksperimen Naive Bayes.ipynb". The first code cell (index 220) displays the first five rows of the Iris dataset using `df.head()`. The output shows columns: Id, SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm, and Species. The second code cell (index 221) uses `iris.isna().sum()` to check for null values, showing zero nulls for all columns. The notebook interface includes a file explorer on the left, a code editor in the center, and a console at the bottom.

```
[220] df.head()
Out[220]:
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0    1             5.1             3.5             1.4             0.1      setosa
1    2             4.9             3.0             1.4             0.1      setosa
2    3             5.4             3.7             1.5             0.2      setosa
3    4             4.3             3.0             1.1             0.1      setosa
4    5             5.0             3.2             1.5             0.2      setosa

[221] iris.isna().sum()
Out[221]:
Id                0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

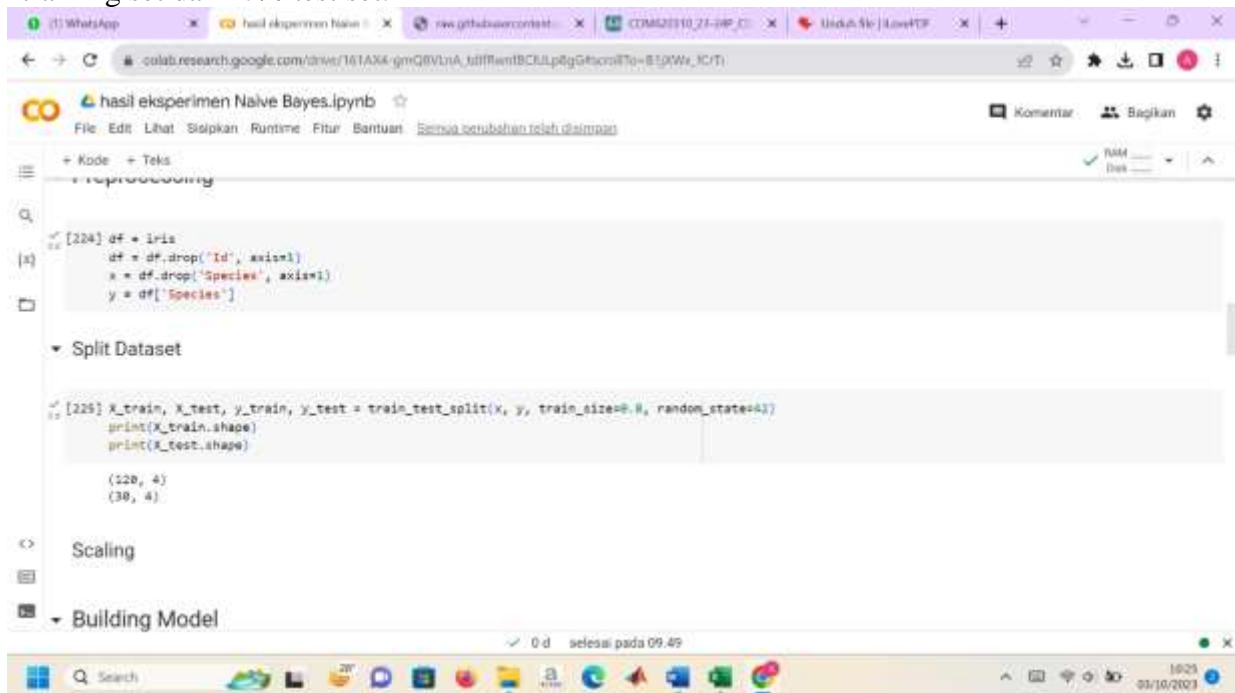
Setelah itu saya menghapus kolom yang menurut saya tidak perlu digunakan yaitu kolom `id` dan menghapus `Species` dari dataframe yang akan menjadi variable dependen.



The screenshot shows the same Jupyter Notebook at a later stage. The third code cell (index 224) performs two operations: dropping the 'Id' column and the 'Species' column from the dataframe. The output shows the updated dataframe structure. The notebook interface is consistent with the previous screenshot, showing the code editor, file explorer, and console.

```
[224] df = iris
df = df.drop('Id', axis=1)
x = df.drop('Species', axis=1)
y = df['Species']
```

Setelah itu saya akan melakukan split data dengan cara membagi dataset menjadi 80% training set dan 40% test set.



The screenshot shows a Google Colab notebook titled "hasil eksperimen Naive Bayes.ipynb". The notebook is divided into sections: "preprocessing", "Split Dataset", "Scaling", and "Building Model". In the "preprocessing" section, the Iris dataset is loaded and the 'Id' and 'Species' columns are dropped. In the "Split Dataset" section, the data is split into training and testing sets using `train_test_split` with a `train_size=0.8` and `random_state=43`. The shapes of the resulting arrays are printed, showing (120, 4) for the training set and (30, 4) for the testing set. The "Scaling" section is currently empty.

```
[224] df = iris
      df = df.drop("Id", axis=1)
      x = df.drop("Species", axis=1)
      y = df["Species"]

▼ Split Dataset

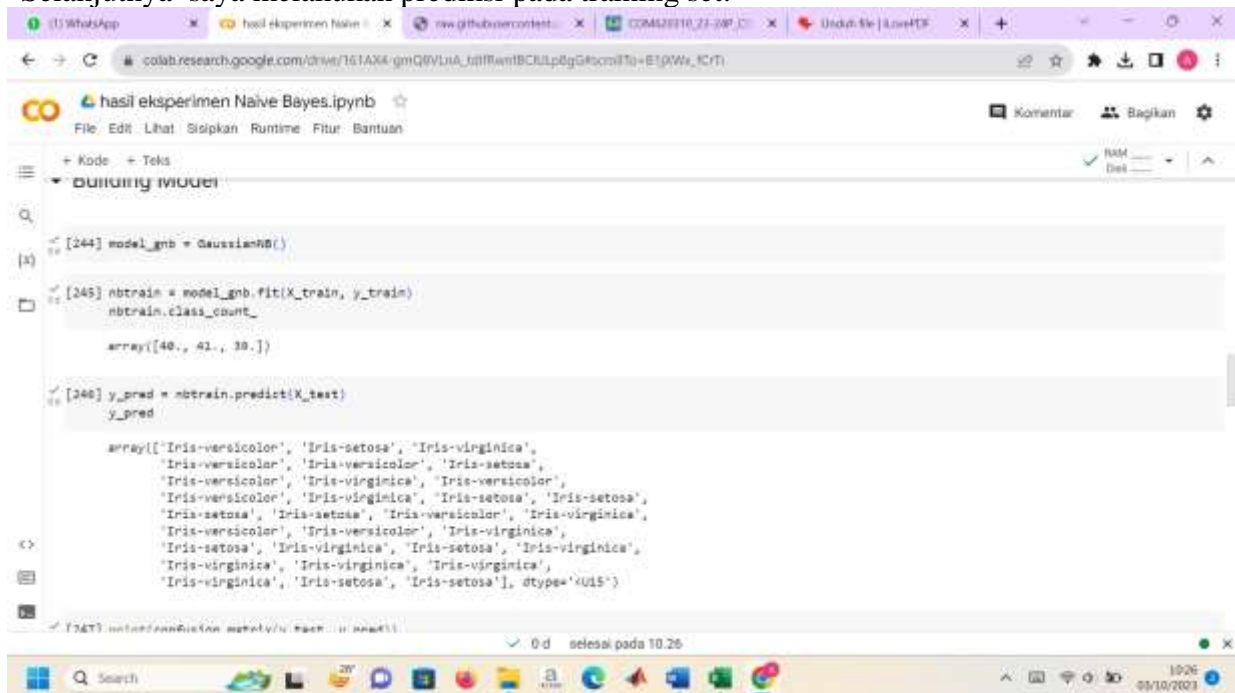
[225] X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.8, random_state=43)
      print(X_train.shape)
      print(X_test.shape)

(120, 4)
(30, 4)

Scaling

▼ Building Model
```

Selanjutnya saya melakukan prediksi pada training set.



The screenshot shows the same Google Colab notebook, now with the "Building Model" section active. In this section, a Gaussian Naive Bayes model is created using `GaussianNB()`. The model is trained on the training data using `nbtrain.fit(X_train, y_train)`, and the class counts are printed, showing an array of [40., 41., 30.]. The model is then used to predict the species of the test set using `nbtrain.predict(X_test)`, and the predicted species are printed, showing an array of predicted species names like 'Iris-versicolor', 'Iris-setosa', and 'Iris-virginica'.

```
[244] model_gnb = GaussianNB()

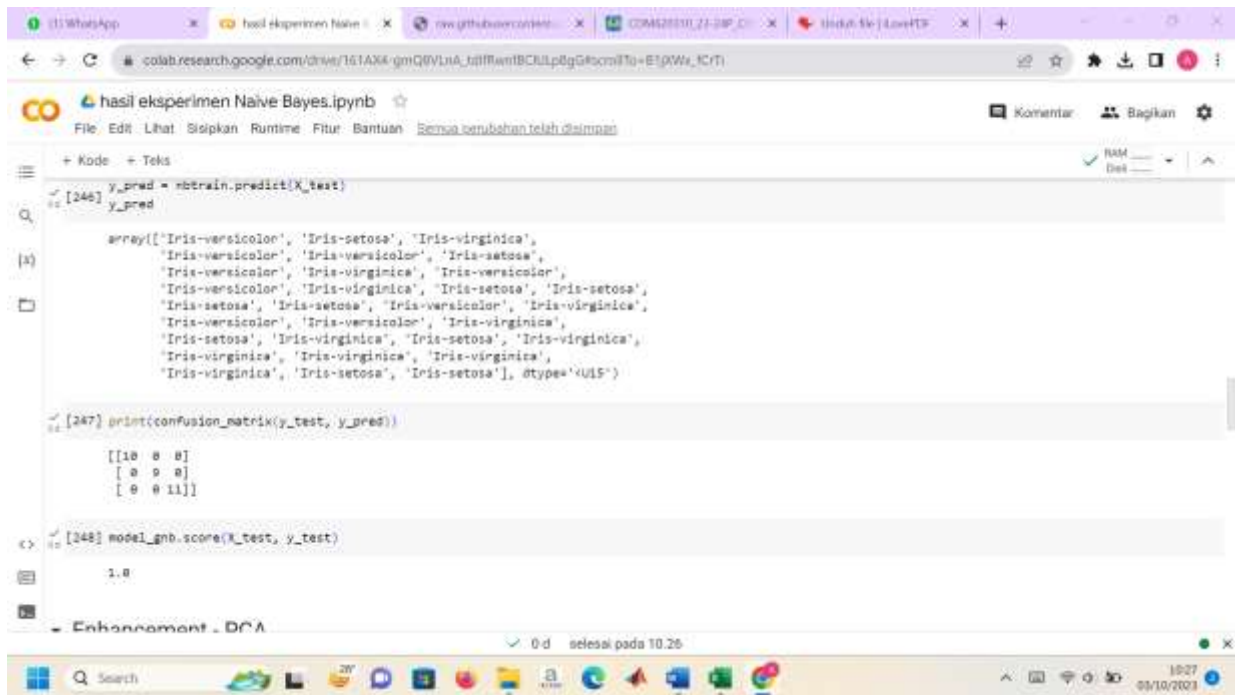
[245] nbtrain = model_gnb.fit(X_train, y_train)
      nbtrain.class_count_

array([40., 41., 30.])

[246] y_pred = nbtrain.predict(X_test)
      y_pred

array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
      'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
      'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
      'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
      'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
      'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
      'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
      'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
      'Iris-virginica', 'Iris-setosa', 'Iris-setosa'], dtype=object)
```

Telah didapatkan  $y_{pred}$ , selanjutnya saya melakukan confusion matrix untuk mengetahui nilai akurasi.



```
[246]: y_pred = nbtrain.predict(X_test)
y_pred

array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa'], dtype=object)

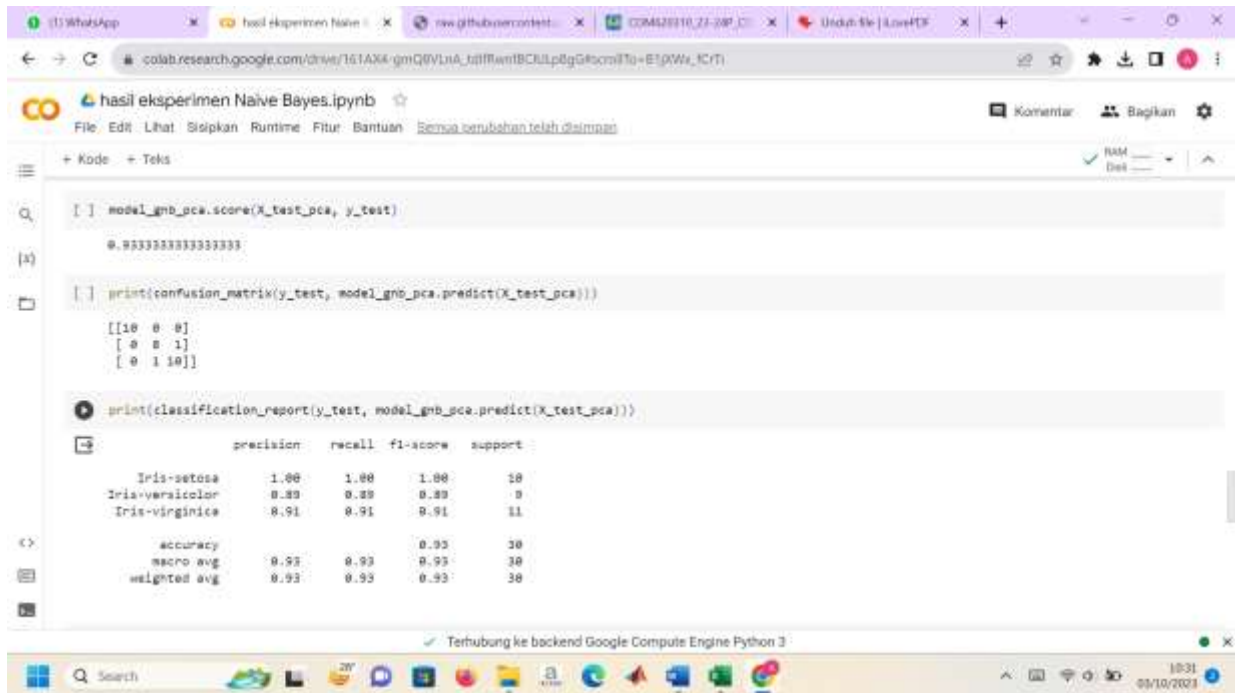
[247]: print(confusion_matrix(y_test, y_pred))

[[10  0  0]
 [ 0  0  0]
 [ 0  0 11]]

[248]: model_gnb.score(X_test, y_test)

1.0
```

Setelah itu saya memakai model pca untuk meningkatkan nilai akurasi sehingga nilai akurasi yang didapatkan adalah 93%.



```
[ ] model_gnb_pca.score(X_test_pca, y_test)

0.9333333333333333

[ ] print(confusion_matrix(y_test, model_gnb_pca.predict(X_test_pca)))

[[10  0  0]
 [ 0  0  1]
 [ 0  1 10]]

[ ] print(classification_report(y_test, model_gnb_pca.predict(X_test_pca)))
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolour	0.00	0.00	0.00	0
Iris-virginica	0.91	0.91	0.91	11
accuracy			0.93	30
macro avg	0.93	0.93	0.93	30
weighted avg	0.93	0.93	0.93	30

Terhubung ke backend Google Compute Engine Python 3