

TUGAS KELOMPOK
PROGRAM STUDI INDEPENDEN
ORBIT FUTURE ACADEMY

Identitas Kelompok

Kelompok	: 10
Nama Anggota	: Halomoan Filipus Simarmata (Jupyter XXI) Diana Eka Riyani (Jupyter XXI) Nyayu Chika Marselina (Jupyter XXI) Sukma Imelda (Cordoba) Athiya Shinta Wulandari (Cordoba)
Coach	: Ipin Sugiyarto
Program	: Foundations of AI and Life Skills for Gen-Z
Hari, Tanggal	: Selasa, 29 Maret 2022

Tugas: Silahkan melakukan Data Preprocessing dengan menggunakan referensi data yg lebih banyak (Sumber bebas silahkan tentukan sendiri). sampai hasil akhir berupa akurasi menggunakan model KNN.

Penyelesaian:

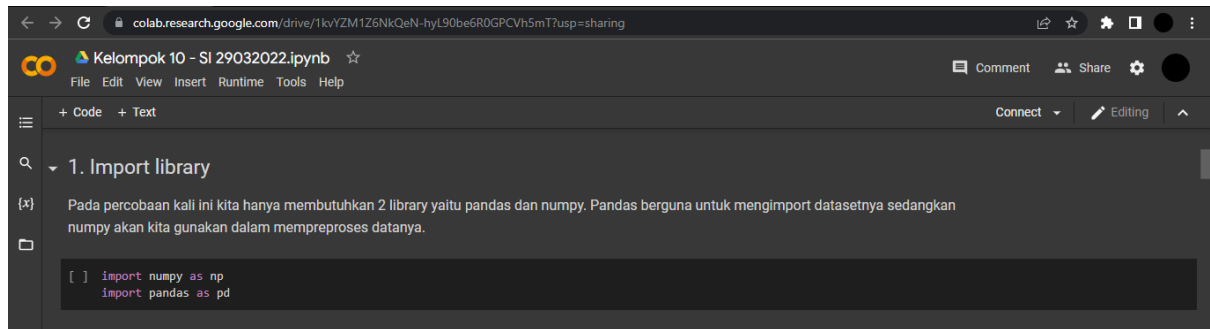
Data Pre-Processing

Why preprocessing?

Data yang kita punya dalam dunia nyata biasanya:

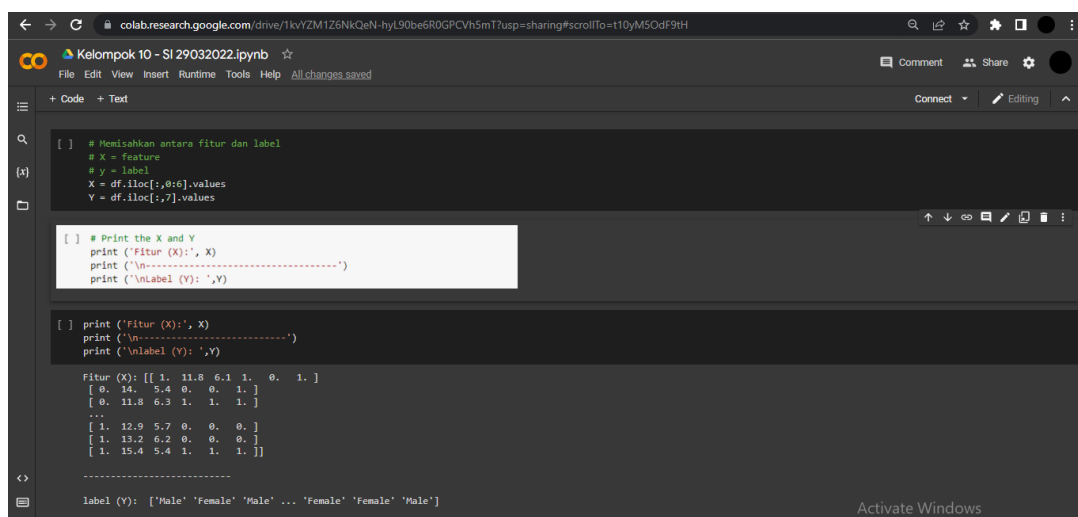
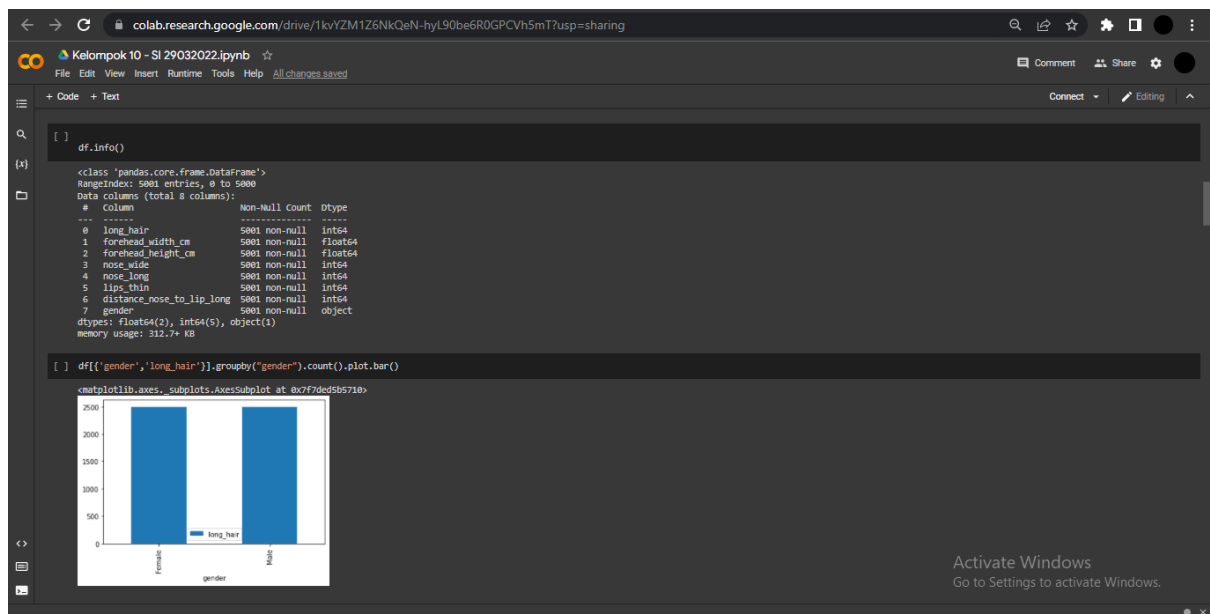
- Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data.
- Noisy: mengandung error seperti missing values dan outliers.
- Inconsistent: mengandung perbedaan dalam penulisan kolom

1. Import Library



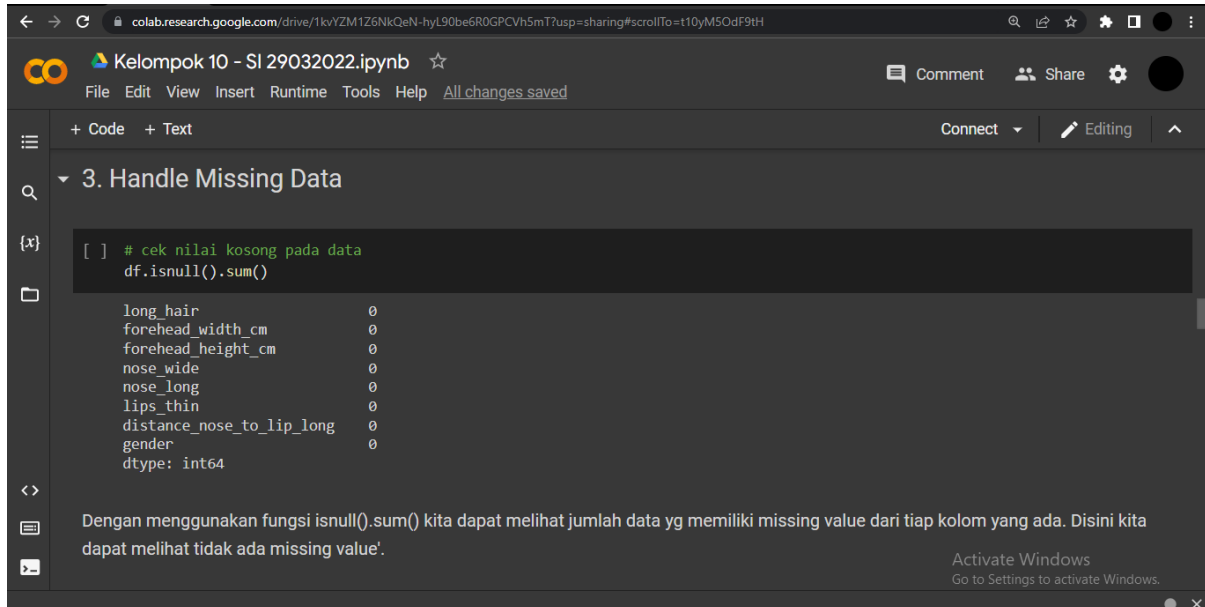
2. Import Dataset

Mengimport dataset yang digunakan, di sini kelompok 10 menggunakan dataset pengklasifikasian gender. Dataset tersebut mengandung 5001 rows dan 8 columns, artinya terdapat 5001 data dan 8 kolom.



3. Handle Missing Data

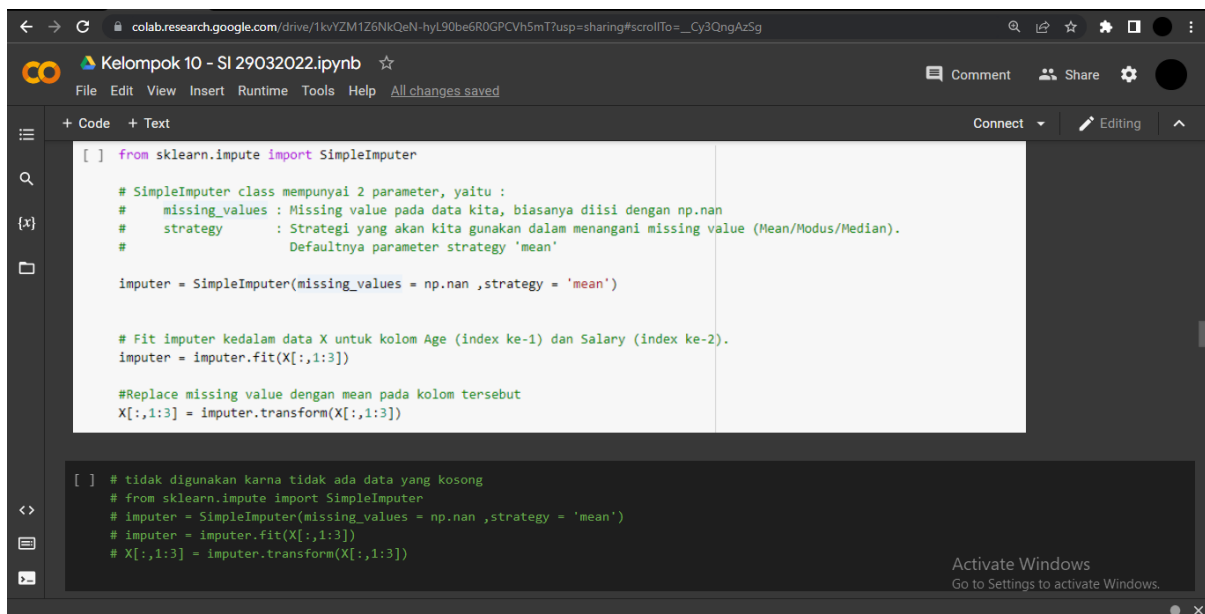
Dengan menggunakan fungsi `isnull.sum()` kita dapat melihat jumlah data yang memiliki missing value dari 8 kolom. Dapat terlihat, tidak ada yang memiliki masalah missing value.



```
[ ] # cek nilai kosong pada data
df.isnull().sum()

long_hair      0
forehead_width_cm  0
forehead_height_cm  0
nose_wide      0
nose_long      0
lips_thin      0
distance_nose_to_lip_long  0
gender         0
dtype: int64
```

Dengan menggunakan fungsi `isnull().sum()` kita dapat melihat jumlah data yg memiliki missing value dari tiap kolom yang ada. Disini kita dapat melihat tidak ada missing value'.



```
[ ] from sklearn.impute import SimpleImputer

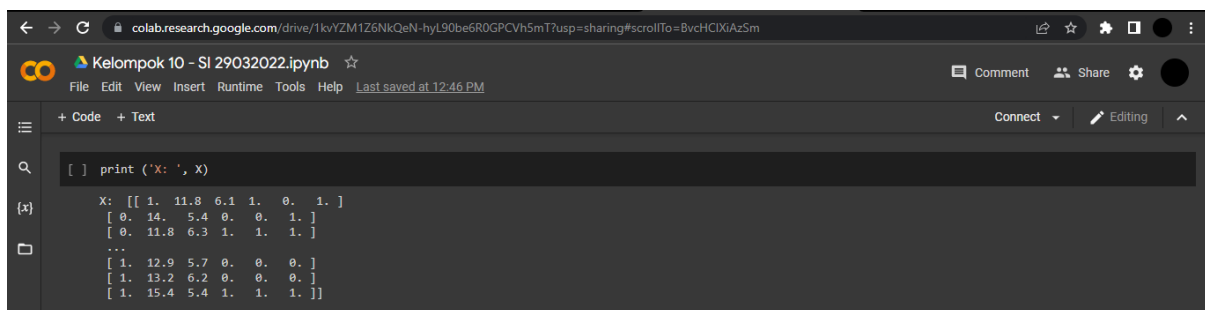
# SimpleImputer class mempunyai 2 parameter, yaitu :
#   missing_values : Missing value pada data kita, biasanya diisi dengan np.nan
#   strategy       : Strategi yang akan kita gunakan dalam menangani missing value (Mean/Modus/Median).
#                   Defaultnya parameter strategy 'mean'

imputer = SimpleImputer(missing_values = np.nan ,strategy = 'mean')

# Fit imputer kedalam data X untuk kolom Age (index ke-1) dan Salary (index ke-2).
imputer = imputer.fit(X[:,1:3])

#Replace missing value dengan mean pada kolom tersebut
X[:,1:3] = imputer.transform(X[:,1:3])

[ ] # tidak digunakan karna tidak ada data yang kosong
# from sklearn.impute import SimpleImputer
# imputer = SimpleImputer(missing_values = np.nan ,strategy = 'mean')
# imputer = imputer.fit(X[:,1:3])
# X[:,1:3] = imputer.transform(X[:,1:3])
```

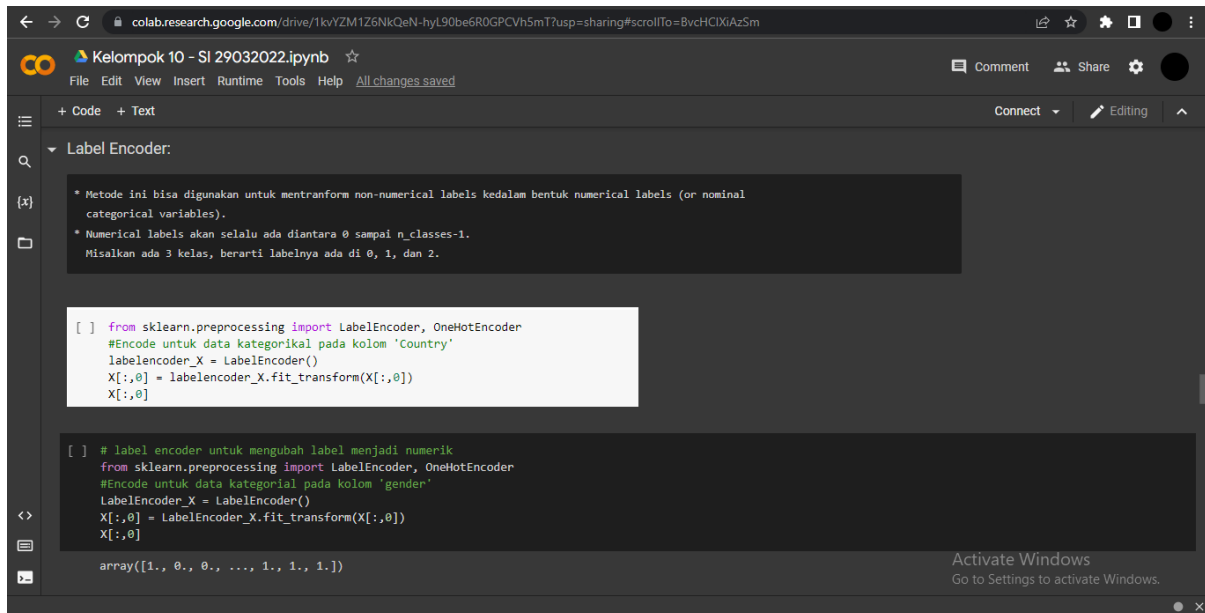


```
[ ] print('X: ', X)

X: [[ 1.  11.8  6.1  1.  0.  1. ]
     [ 0.  14.   5.4  0.  0.  1. ]
     [ 0.  11.8  6.3  1.  1.  1. ]
     ...
     [ 1.  12.9  5.7  0.  0.  0. ]
     [ 1.  13.2  6.2  0.  0.  0. ]
     [ 1.  15.4  5.4  1.  1.  1. ]]
```

4. Encode the Categorical data

Categorical data biasanya memiliki tipe data string. Beberapa algoritma dapat memproses categorical data yang berbentuk string secara langsung seperti decision tree. Akan tetapi, banyak machine learning algorithms tidak dapat memproses categorical data yang berbentuk string secara langsung. Mereka membutuhkan variabel yang berbentuk numeric. Itu artinya categorical data harus dikonversi bentuknya ke dalam numerical form.

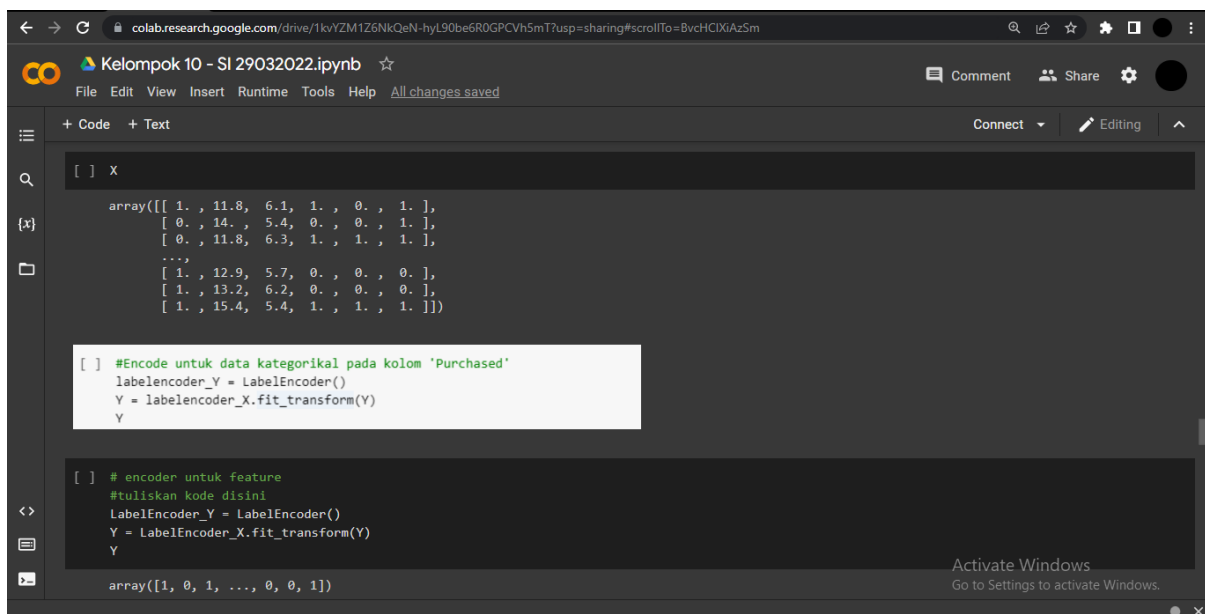


```
colab.research.google.com/drive/1kvYZM1Z6NkQeN-hyL90be6R0GPCVh5mT?usp=sharing#scrollTo=BvcHCIXGAz5m
Kelompok 10 - SI 29032022.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Label Encoder:
* Metode ini bisa digunakan untuk mentransform non-numerical labels kedalam bentuk numerical labels (or nominal categorical variables).
* Numerical labels akan selalu ada diantara 0 sampai n_classes-1.
Misalkan ada 3 kelas, berarti labelnya ada di 0, 1, dan 2.

[ ] from sklearn.preprocessing import LabelEncoder, OneHotEncoder
#Encode untuk data kategorikal pada kolom 'Country'
labelencoder_X = LabelEncoder()
X[:,0] = labelencoder_X.fit_transform(X[:,0])
X[:,0]

[ ] # label encoder untuk mengubah label menjadi numerik
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
#Encode untuk data kategorikal pada kolom 'gender'
LabelEncoder_X = LabelEncoder()
X[:,0] = LabelEncoder_X.fit_transform(X[:,0])
X[:,0]

array([1., 0., 0., ..., 1., 1., 1.])
Activate Windows
Go to Settings to activate Windows.
```



```
colab.research.google.com/drive/1kvYZM1Z6NkQeN-hyL90be6R0GPCVh5mT?usp=sharing#scrollTo=BvcHCIXGAz5m
Kelompok 10 - SI 29032022.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[ ] X
array([[ 1., 11.8,  6.1,  1.,  0.,  1. ],
       [ 0., 14.,  5.4,  0.,  0.,  1. ],
       [ 0., 11.8,  6.3,  1.,  1.,  1. ],
       ...,
       [ 1., 12.9,  5.7,  0.,  0.,  0. ],
       [ 1., 13.2,  6.2,  0.,  0.,  0. ],
       [ 1., 15.4,  5.4,  1.,  1.,  1. ]])

[ ] #Encode untuk data kategorikal pada kolom 'Purchased'
labelencoder_Y = LabelEncoder()
Y = labelencoder_X.fit_transform(Y)
Y

[ ] # encoder untuk feature
#tuliskan kode disini
LabelEncoder_Y = LabelEncoder()
Y = LabelEncoder_X.fit_transform(Y)
Y

array([1, 0, 1, ..., 0, 0, 1])
Activate Windows
Go to Settings to activate Windows.
```

5. Splitting Dataset

```
[ ] from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2,random_state = 0)

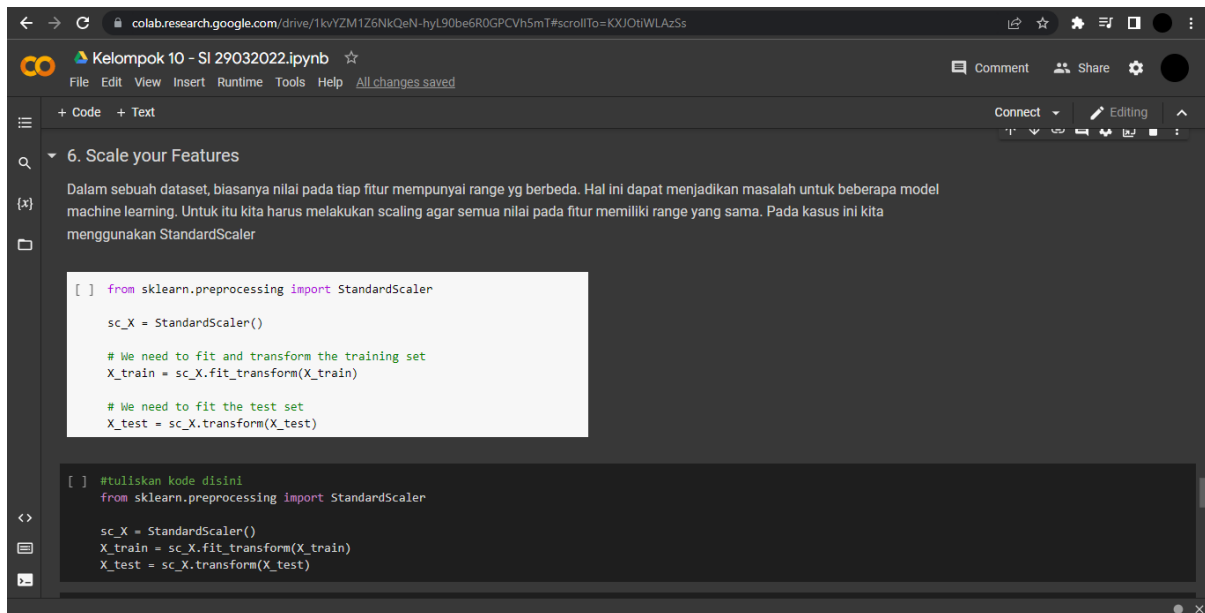
[ ] #tuliskan kode disini
# untuk membagi data test dan train, data test diberi sebesar 30%
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.3, random_state=0)
```

```
[ ] # Print the shape of the dataset
print ('X_train:', (str(X_train.shape)))
print ('-----')
print ('X_test:', (str(X_test.shape)))
print ('-----')
print ('Y_train:', (str(Y_train.shape)))
print ('-----')
print ('Y_test:', (str(Y_test.shape)))
print ('-----')

X_train: (3500, 6)
-----
X_test: (1501, 6)
-----
Y_train: (3500,)
-----
Y_test: (1501,)
```

6. Scale your Features

Dalam sebuah dataset, biasanya nilai pada tiap fitur mempunyai range yang berbeda. Hal ini dapat menjadikan masalah untuk beberapa model machine learning. Untuk itu kita harus melakukan scaling agar semua nilai pada fitur memiliki range yang sama. Pada kasus ini kita menggunakan StandardScaler



The screenshot shows a Google Colab notebook titled "Kelompok 10 - SI 29032022.ipynb". The notebook is in the "Code" tab. The first code cell contains the following Python code:

```
[ ] from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()

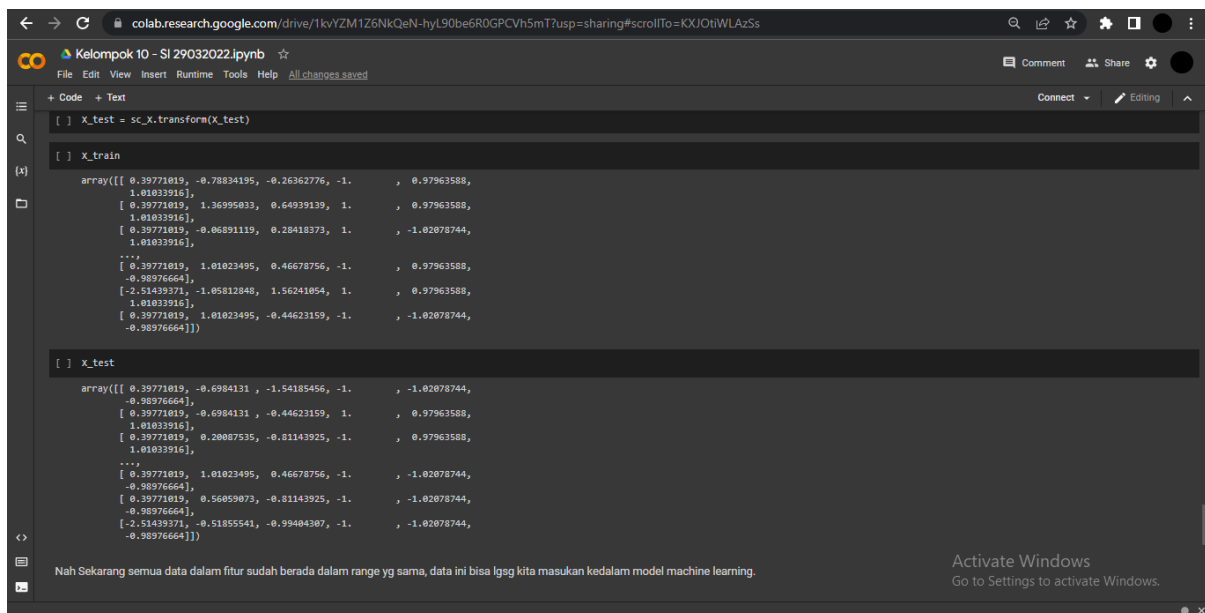
# We need to fit and transform the training set
X_train = sc_X.fit_transform(X_train)

# We need to fit the test set
X_test = sc_X.transform(X_test)
```

The second code cell contains the following Python code:

```
[ ] #tuliskan kode disini
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```



The screenshot shows the same Google Colab notebook, but now the output of the transformation is visible. The first code cell's output is:

```
[ ] X_test = sc_X.transform(X_test)
```

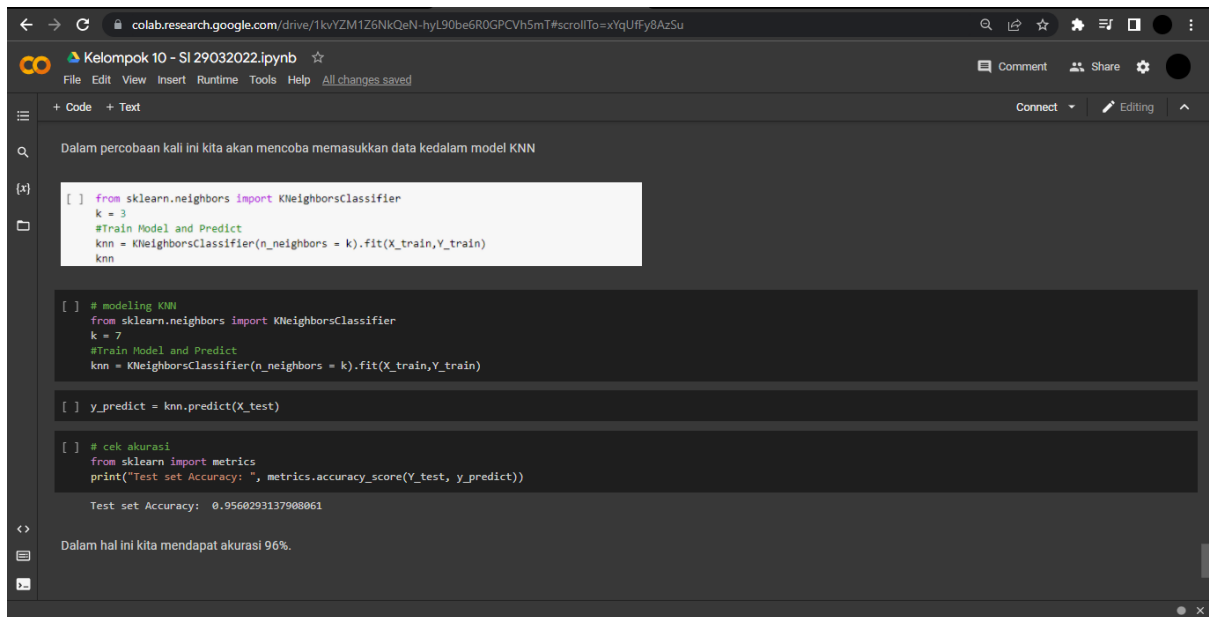
The second code cell's output is:

```
[ ] X_train
array([[ 0.39771019, -0.78834195, -0.26362776, -1.         ,  0.97963588,
        1.01033916],
       [ 0.39771019,  1.36995033,  0.64939139,  1.         ,  0.97963588,
        1.01033916],
       [ 0.39771019, -0.06891119,  0.29418373,  1.         , -1.02078744,
        1.01033916],
       ...,
       [ 0.39771019,  1.01023495,  0.46678756, -1.         ,  0.97963588,
       -0.98976664],
       [-2.51439371, -1.05812848,  1.56241854,  1.         ,  0.97963588,
        1.01033916],
       [ 0.39771019,  1.01023495, -0.44623159, -1.         , -1.02078744,
       -0.98976664]])
```

The third code cell's output is:

```
[ ] X_test
array([[ 0.39771019, -0.6984131 , -1.54185456, -1.         , -1.02078744,
       -0.98976664],
       [ 0.39771019, -0.6984131 , -0.44623159,  1.         ,  0.97963588,
        1.01033916],
       [ 0.39771019,  0.20087535, -0.81143925, -1.         ,  0.97963588,
        1.01033916],
       ...,
       [ 0.39771019,  1.01023495,  0.46678756, -1.         , -1.02078744,
       -0.98976664],
       [ 0.39771019,  0.56059873, -0.81143925, -1.         , -1.02078744,
       -0.98976664],
       [-2.51439371, -0.51855541, -0.99404307, -1.         , -1.02078744,
       -0.98976664]])
```

Nah Sekarang semua data dalam fitur sudah berada dalam range yg sama, data ini bisa lgsg kita masukan kedalam model machine learning.



```
[ ] from sklearn.neighbors import KNeighborsClassifier
k = 3
#Train Model and Predict
knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,Y_train)
knn

[ ] # modeling KNN
from sklearn.neighbors import KNeighborsClassifier
k = 7
#Train Model and Predict
knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,Y_train)

[ ] y_predict = knn.predict(X_test)

[ ] # cek akurasi
from sklearn import metrics
print("Test set Accuracy: ", metrics.accuracy_score(Y_test, y_predict))

Test set Accuracy: 0.9560293137908061

Dalam hal ini kita mendapat akurasi 96%.
```

Diperoleh nilai akurasi maksimal pada nilai $k = 7$ senilai 96%.