Nama/NIM Marah nafiah M./ E42240962 Adzra Syafa K./E42241974 Savana Farel A/E42240035 Devina Putri I./E422440721 Dewi Suciyo W./E42240571

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns
```

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df = pd.read_csv("/content/diabetes.csv")
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Langkah berikutnya: Buat kode dengan df | New interactive sheet

```
df.info()

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
```

```
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
df['Outcome'].value_counts()
```

Outcome	count
0	500
1	268

dtype: int64

```
correlation = df.corr()

correlation['Outcome'].sort_values(ascending=False)
```

	Outcome
Outcome	1.000000
Glucose	0.466581
BMI	0.292695
Age	0.238356
Pregnancies	0.221898
DiabetesPedigreeFunction	0.173844
Insulin	0.130548
SkinThickness	0.074752
BloodPressure	0.065068

dtype: float64

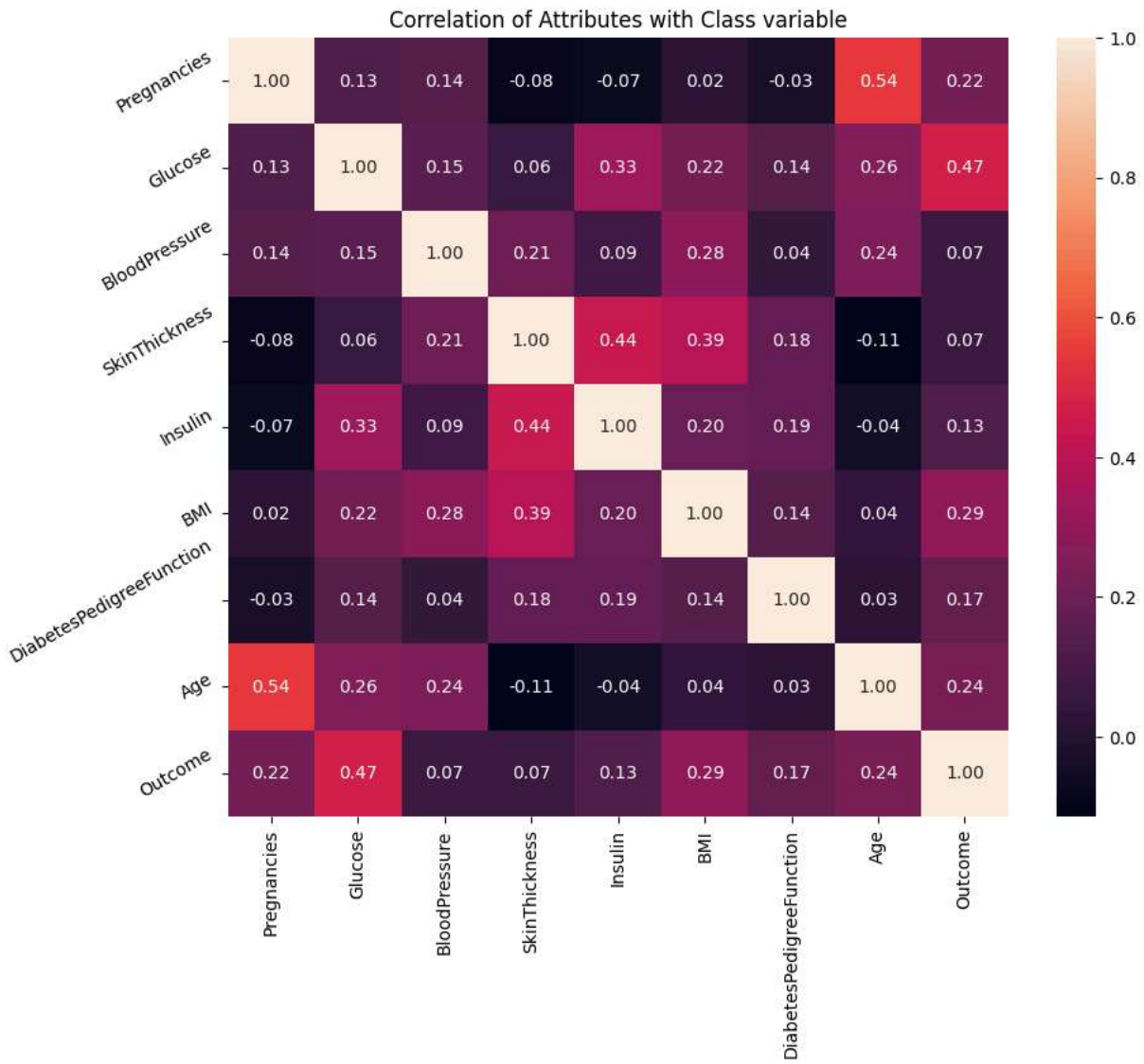
```
correlation = df.corr()

correlation['Outcome'].sort_values(ascending=False)
```

	Outcome
Outcome	1.000000
Glucose	0.466581
BMI	0.292695
Age	0.238356
Pregnancies	0.221898
DiabetesPedigreeFunction	0.173844
Insulin	0.130548
SkinThickness	0.074752
BloodPressure	0.065068

dtype: float64

```
plt.figure(figsize=(10,8))

plt.title('Correlation of Attributes with Class variable')

a = sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='white')

a.set_xticklabels(a.get_xticklabels(), rotation=90)
```

```
a.set_yticklabels(a.get_yticklabels(), rotation=30)

plt.show()
```




```
X = df.drop(['Outcome'], axis=1)

y = df['Outcome']
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X = pd.DataFrame(scaler.fit_transform(X),columns=X.columns)

X
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.639947	0.848324	0.149641	0.907270	-0.692891	0.204013	0.468492	1.425995
1	-0.844885	-1.123396	-0.160546	0.530902	-0.692891	-0.684422	-0.365061	-0.190672
2	1.233880	1.943724	-0.263941	-1.288212	-0.692891	-1.103255	0.604397	-0.105584
3	-0.844885	-0.998208	-0.160546	0.154533	0.123302	-0.494043	-0.920763	-1.041549
4	-1.141852	0.504055	-1.504687	0.907270	0.765836	1.409746	5.484909	-0.020496
...	...	...	...	...	...	...	...	...
763	1.827813	-0.622642	0.356432	1.722735	0.870031	0.115169	-0.908682	2.532136
764	-0.547919	0.034598	0.046245	0.405445	-0.692891	0.610154	-0.398282	-0.531023
765	0.342981	0.003301	0.149641	0.154533	0.279594	-0.735190	-0.685193	-0.275760
766	-0.844885	0.159787	-0.470732	-1.288212	-0.692891	-0.240205	-0.371101	1.170732
767	-0.844885	-0.873019	0.046245	0.656358	-0.692891	-0.202129	-0.473785	-0.871374

768 rows × 8 columns

Langkah berikutnya: Buat kode dengan X New interactive sheet

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.21115, random_state = 0)

X_train.shape, X_test.shape
```

```
((605, 8), (163, 8))
```

```
X_train
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
97	-0.844885	-1.561556	-1.091105	-0.159107	-0.032990	-1.471321	-0.449624	-0.956462
530	-0.547919	0.034598	-0.470732	-0.159107	0.227496	-0.278280	0.740303	-0.956462
327	1.827813	1.818535	0.046245	-1.288212	-0.692891	0.394392	-0.821099	0.319855
619	-1.141852	-0.059293	-3.572597	-1.288212	-0.692891	0.051710	-0.999286	-0.786286
518	2.718712	-1.405071	-0.470732	-1.288212	-0.692891	0.102477	-0.881501	0.660206
...	...	...	...	...	...	...	...	...
763	1.827813	-0.622642	0.356432	1.722735	0.870031	0.115169	-0.908682	2.532136
192	0.936914	1.192592	-0.160546	-1.288212	-0.692891	-0.202129	-0.268417	0.234767
629	0.046014	-0.841722	-0.212243	0.091805	-0.692891	-0.925569	-0.978145	-1.041549
559	2.124780	-1.123396	0.253036	-1.288212	-0.692891	-0.240205	-0.519087	0.149679
684	0.342981	0.472758	0.666618	-1.288212	-0.692891	-4.060474	0.507754	3.042663

606 rows × 8 columns

Langkah berikutnya: Buat kode dengan X_train New interactive sheet

```
X_test
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
661	-0.844885	2.444478	0.356432	1.409094	-0.692891	1.384362	2.784923	-0.956462
122	-0.547919	-0.434859	0.253036	0.593630	0.175399	0.204013	-0.204994	-0.871374
113	0.046014	-1.405071	-0.367337	-1.288212	-0.692891	0.254780	-0.244256	-0.701198
14	0.342981	1.411672	0.149641	-0.096379	0.826616	-0.785957	0.347687	1.511083
529	-1.141852	-0.309671	-0.212243	-1.288212	-0.692891	-0.938260	0.568156	-0.190672
...	...	...	...	...	...	...	...	...
319	0.639947	2.287992	0.459827	-1.288212	-0.692891	-1.077872	-1.035527	2.191785
						032990	-1.471321	-0.449624	-0.956462

153 rows × 8 columns

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=5,weights='distance',metric='euclidean')

knn.fit(X_train, y_train)
```

KNeighborsClassifier

```
KNeighborsClassifier(metric='euclidean', weights='distance')
```

```
y_pred = knn.predict(X_test)

y_pred
```

```
array([1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0])
```

```
from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score: 0.8098
```

```
y_pred_train = knn.predict(X_train)

print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train)))
```

```
Training-set accuracy score: 1.0000
```

```
print('Training set score: {:.4f}'.format(knn.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(knn.score(X_test, y_test)))
```

```
Training set score: 1.0000
Test set score: 0.8098
```

```
y_test.value_counts()
```

Outcome	count
0	111
1	52

dtype: int64

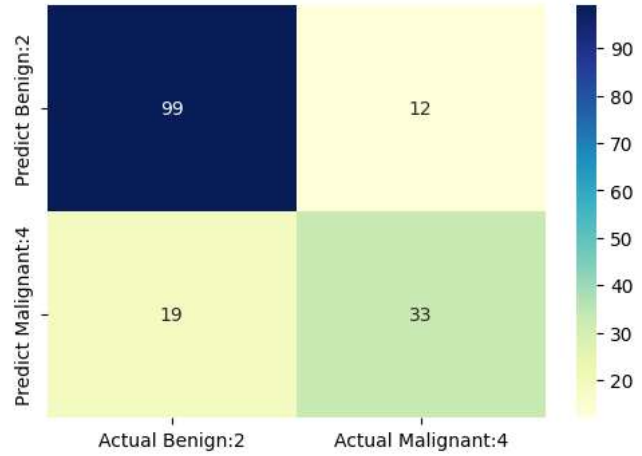
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Negatives(TP) = ', cm[0,0])

print('\nTrue Positives(TN) = ', cm[1,1])
```

```
print('\nFalse Negatives(FP) = ', cm[0,1])
```

```
Confusion matrix

 [[99 12]
 [19 33]]

True Negatives(TP) =  99

True Positives(TN) =  33

False Negatives(FP) =  12

False Positives(FN) =  19
```

```
plt.figure(figsize=(6,4))

cm_matrix = pd.DataFrame(cm, columns=['Actual Benign:2', 'Actual Malignant:4'], index=['Predict Benign:2', 'Predict Malignant:4'

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

```
<Axes: >
```




```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.89      0.86       111
           1       0.73      0.63      0.68        52

    accuracy                           0.81       163
   macro avg       0.79      0.76      0.77       163
weighted avg       0.81      0.81      0.81       163
```

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(knn, X, y, cv = 10, scoring='accuracy')

print('Cross-validation scores:{}'.format(scores))
```

```
Cross-validation scores:[0.64935065 0.77922078 0.71428571 0.71428571 0.71428571 0.75324675
 0.72727273 0.85714286 0.69736842 0.75      ]
```

```
print('Average cross-validation score: {:.2f}'.format(scores.mean()))
```

```
Average cross-validation score: 0.74
```