1. Reva Dira Putri (E42241593) Gol B
2. Maria Maksanlina M (E42240119) Gol A
3. Mutiara Ramadhani (E42241957) Gol B
4. Revinka Yos Sheyla (E42241640) Gol B

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files diabetes.csv

diabetes.csv(text/csv) - 23873 bytes, last modified: 11/3/2025 - 100% done

```
Saving diabetes.csv to diabetes (2).csv
```

```
import pandas as pd
data = pd.read_csv('diabetes.csv')
data.head()
```

	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

Next steps: Generate code with data | New interactive sheet

```
data.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
```

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

Outcome	count
0	500
1	268

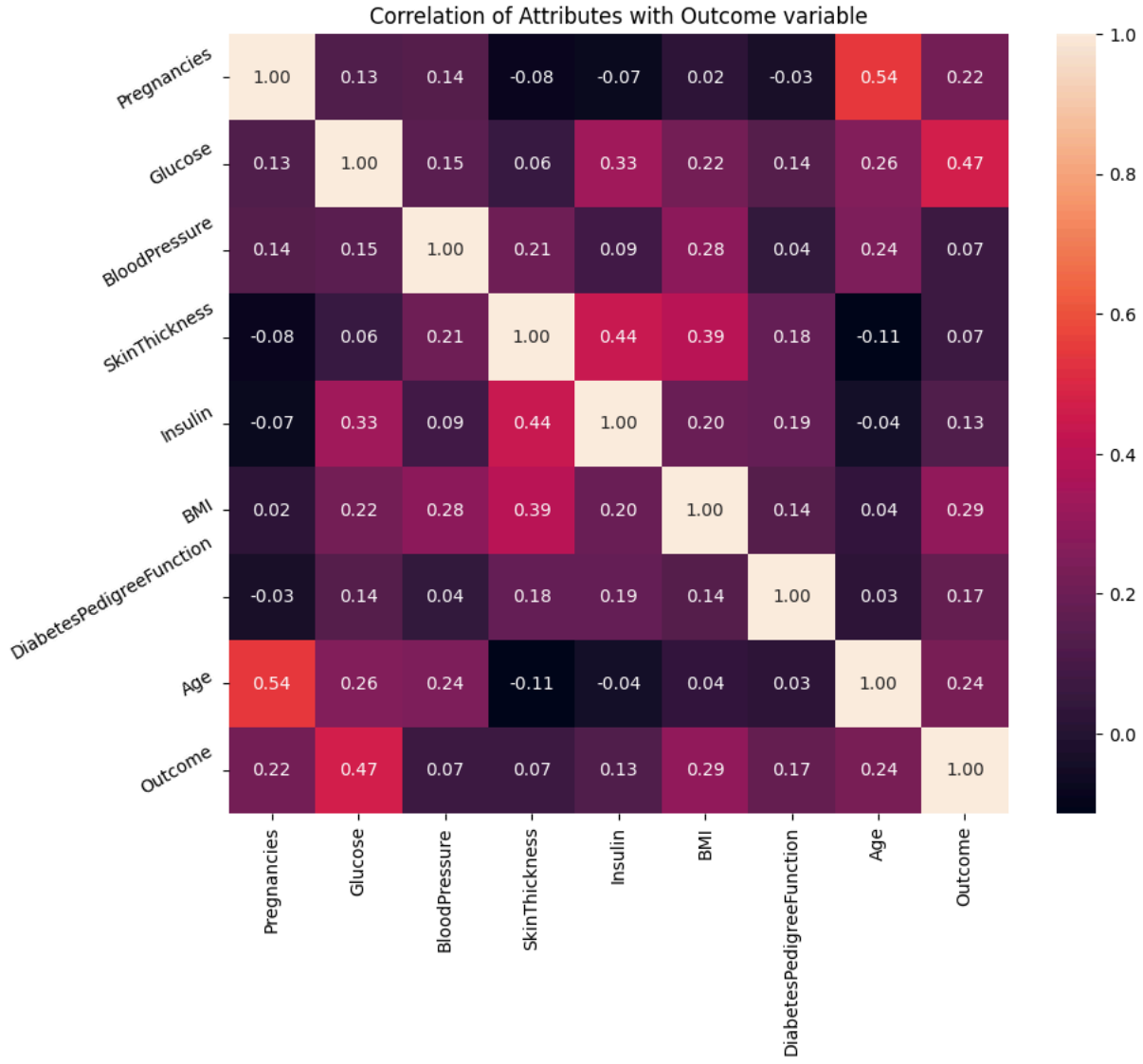
dtype: int64

```
correlation = data.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 Outcome 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 = data.drop(['Outcome'], axis=1)
y = data['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

Next steps: Generate code with 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 = 0.15, random_state = 0)
X_train.shape, X_test.shape
```

```
((652, 8), (116, 8))
```

```
X_train
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
279	-0.547919	-0.403562	-0.367337	-0.660932	1.720954	-0.849417	1.235602	-0.956462
258	-0.844885	2.256695	-0.987710	-0.284563	2.563195	-0.773265	0.553055	-0.786286
249	-0.844885	-0.309671	0.873409	-0.096379	-0.692891	-0.240205	-0.993245	-0.871374
740	2.124780	-0.027996	0.563223	1.032726	0.609544	1.308210	0.945671	1.255820
725	0.046014	-0.278373	0.459827	1.220910	-0.692891	0.940144	-0.712374	0.404942
...	...	...	...	...	...	...	...	...
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

652 rows × 8 columns

Next steps: Generate code with 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
...	...	...	...	...	...	...	...	...
214	1.530847	-0.278373	0.666618	0.719086	0.826616	0.280164	-0.639892	0.234767
586	1.233880	0.691838	-0.160546	-1.288212	-0.692891	0.369008	-1.035527	0.660206
187	-0.844885	0.222381	1.493782	1.283638	-0.189283	0.000942	2.564454	-0.020496
415	-0.250952	1.630752	0.770014	0.781814	3.422802	0.470543	-0.645932	-0.956462
283	0.936914	1.255187	0.873409	-1.288212	-0.692891	-0.202129	-0.926803	1.170732

116 rows × 8 columns

Next steps: Generate code with X_test | New interactive sheet

```
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, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 1])
```

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

```
Model accuracy score: 0.8190
```

```
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: 0.8190
```

```
y_test.value_counts()
```

Outcome	count
0	78
1	38

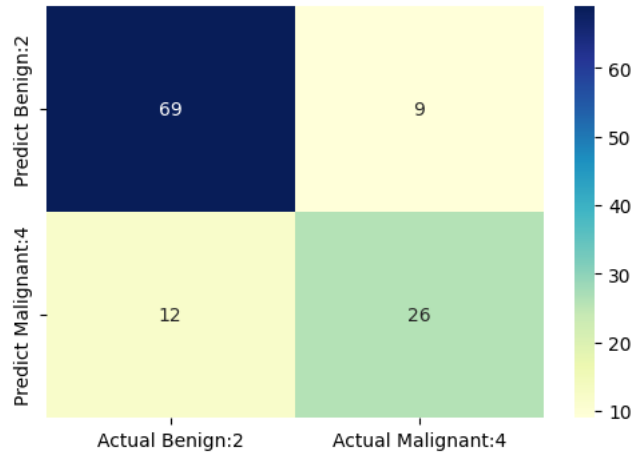
dtype: int64

```
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])
print('\nFalse Positives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[69  9]
 [12 26]]

True Negatives(TP) =  69

True Positives(TN) =  26

False Negatives(FP) =  9

False Positives(FN) =  12
```

```
plt.figure(figsize=(6,4))
cm_matrix = pd.DataFrame(cm, columns=['Actual Benign:2', 'Actual Malignant:4'], index=['Predict Benign:2', 'Predict Malignа
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       0.85      0.88      0.87        78
           1       0.74      0.68      0.71        38

    accuracy                           0.82       116
   macro avg       0.80      0.78      0.79       116
weighted avg       0.82      0.82      0.82       116
```

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(knn, X, y, cv = 10, scoring='accuracy')
```