For columns:

Glucose,
BloodPressure,
BMI,
DiabetesPedigreeFunction

If the column value is 0, then they should be considered as **missing data.**

1. Firstly, replace all Missing values with relevant figures.

```
import numpy as np
import pandas as pd
df = pd.read_csv('Dataset_Day7.csv')
print(df.info())
missing_value_percent = df.isna().sum() / len(df)
* 100
print(missing_value_percent)
skewness = df.skew()
print(skewness)
df["Glucose"].fillna(df["Glucose"].median(),
inplace=True)
df["BloodPressure"].fillna(df["BloodPressure"].me
an(), inplace=True)
df["BMI"].fillna(df["BMI"].median(),
inplace=True)
df["Outcome"].fillna(df["Outcome"].mean(),
inplace=True)
print(df.info())
```

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day12Q1.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 7 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   BMI                       768 non-null    float64
 4   DiabetesPedigreeFunction  768 non-null    float64
 5   Age                       768 non-null    int64
 6   Outcome                   768 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 42.1 KB
None
Pregnancies                 0.0
Glucose                     0.0
BloodPressure               0.0
BMI                         0.0
DiabetesPedigreeFunction    0.0
Age                         0.0
Outcome                     0.0
dtype: float64
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 7 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   BMI                       768 non-null    float64
 4   DiabetesPedigreeFunction  768 non-null    float64
 5   Age                       768 non-null    int64
 6   Outcome                   768 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 42.1 KB
None

Process finished with exit code 0
```

2.Then remove all existing outliers and get the final data for classification

```python
import numpy as np
import pandas as pd
df = pd.read_csv('Dataset_Day7.csv')
print(df.info())
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
```

```
skewness = df.skew()
print(skewness)
df["Glucose"].fillna(df["Glucose"].median(),
inplace=True)
df["BloodPressure"].fillna(df["BloodPressure"].mean
(), inplace=True)
df["BMI"].fillna(df["BMI"].median(), inplace=True)
df["Outcome"].fillna(df["Outcome"].mean(),
inplace=True)
print(df.info())
OutlierData = pd.DataFrame()
temp = df[["Pregnancies", "Glucose",
"BloodPressure","BMI","DiabetesPedigreeFunction"]]
for col in ["Pregnancies", "Glucose",
"BloodPressure","BMI","DiabetesPedigreeFunction"]:
    Q1 = temp[col].quantile(0.25)  # Gives 25th
Percentile or Q1
    Q3 = temp[col].quantile(0.75)  # Gives 75th
Percentile or Q3

    IQR = Q3 - Q1

    UpperBound = Q3 + 1.5 * IQR
    LowerBound = Q1 - 1.5 * IQR

    OutlierData[col] = temp[col][(temp[col] <
LowerBound) | (temp[col] > UpperBound)]
    print(len(OutlierData))
    df_OutlierFree = df.drop(OutlierData.index,
axis=0)
    df_OutlierFree.info()
```

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day12Q2.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 7 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   BMI                       768 non-null    float64
 4   DiabetesPedigreeFunction  768 non-null    float64
 5   Age                       768 non-null    int64
 6   Outcome                   768 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 42.1 KB
None
Pregnancies                 0.0
Glucose                     0.0
BloodPressure               0.0
BMI                         0.0
DiabetesPedigreeFunction    0.0
Age                         0.0
Outcome                     0.0
dtype: float64
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 7 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   BMI                       768 non-null    float64
 4   DiabetesPedigreeFunction  768 non-null    float64
 5   Age                       768 non-null    int64
 6   Outcome                   768 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 42.1 KB
None
4
<class 'pandas.core.frame.DataFrame'>
Index: 764 entries, 0 to 767
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               764 non-null    int64
 1   Glucose                   764 non-null    int64
 2   BloodPressure             764 non-null    int64
 3   BMI                       764 non-null    float64
 4   DiabetesPedigreeFunction  764 non-null    float64
```

```
 5   Age                       764 non-null    int64
 6   Outcome                   764 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 47.8 KB
4
<class 'pandas.core.frame.DataFrame'>
Index: 764 entries, 0 to 767
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               764 non-null    int64
 1   Glucose                   764 non-null    int64
 2   BloodPressure             764 non-null    int64
 3   BMI                       764 non-null    float64
 4   DiabetesPedigreeFunction  764 non-null    float64
 5   Age                       764 non-null    int64
 6   Outcome                   764 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 47.8 KB
4
<class 'pandas.core.frame.DataFrame'>
Index: 764 entries, 0 to 767
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               764 non-null    int64
 1   Glucose                   764 non-null    int64
 2   BloodPressure             764 non-null    int64
 3   BMI                       764 non-null    float64
 4   DiabetesPedigreeFunction  764 non-null    float64
 5   Age                       764 non-null    int64
 6   Outcome                   764 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 47.8 KB
4
<class 'pandas.core.frame.DataFrame'>
Index: 764 entries, 0 to 767
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               764 non-null    int64
 1   Glucose                   764 non-null    int64
 2   BloodPressure             764 non-null    int64
 3   BMI                       764 non-null    float64
 4   DiabetesPedigreeFunction  764 non-null    float64
 5   Age                       764 non-null    int64
 6   Outcome                   764 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 47.8 KB
4
<class 'pandas.core.frame.DataFrame'>
Index: 764 entries, 0 to 767
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               764 non-null    int64
 1   Glucose                   764 non-null    int64
 2   BloodPressure             764 non-null    int64
 3   BMI                       764 non-null    float64
```

File - Day12Q2

```
118  4   DiabetesPedigreeFunction  764 non-null    float64
119  5   Age                       764 non-null    int64
120  6   Outcome                   764 non-null    int64
121 dtypes: float64(2), int64(5)
122 memory usage: 47.8 KB
123
124 Process finished with exit code 0
125
```

3.Split the data into 80% training and 20% testing data. Use target variable as 'Outcome'.

a. Use Naïve Bayes Classifier algorithm to classify *Outcome* and print the default model performance metrics: Accuracy, Precision, Recall, F1Score.

b. Use k-Fold Cross Validation to find the best model performance metrics .

c. Try out Adaboost algorithm for Naïve Bayes Classifier improvement. [Optional]

```
import numpy as np
import pandas as pd
from sklearn.model_selection import
train_test_split, cross_val_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,
precision_score, recall_score, f1_score
from sklearn.ensemble import AdaBoostClassifier

# Load the dataset
df = pd.read_csv('Dataset_Day7.csv')

# Handle missing values
df["Glucose"].fillna(df["Glucose"].median(),
inplace=True)
df["BloodPressure"].fillna(df["BloodPressure"].me
an(), inplace=True)
df["BMI"].fillna(df["BMI"].median(),
inplace=True)
df["Outcome"].fillna(df["Outcome"].mean(),
inplace=True)

# Split the data into training and testing sets
X = df.drop('Outcome', axis=1)
```

```
y = df['Outcome']  # Target variable

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=50)
# a. Naive Bayes Classifier
nb_clf = GaussianNB()
nb_clf.fit(X_train, y_train)
y_pred = nb_clf.predict(X_test)

print("Naive Bayes Classifier - Default Model
Performance Metrics:")
print("Accuracy: ", accuracy_score(y_test,
y_pred))
print("Precision: ", precision_score(y_test,
y_pred))
print("Recall: ", recall_score(y_test, y_pred))
print("F1-Score: ", f1_score(y_test, y_pred))

# b. k-Fold Cross Validation
nb_clf_cv = GaussianNB()
cv_scores = cross_val_score(nb_clf_cv, X, y,
cv=5, scoring='f1')

print("Naive Bayes Classifier - Cross-Validated
F1-Score:", np.mean(cv_scores))

# c. Adaboost with Naive Bayes
adaboost_nb_clf =
AdaBoostClassifier(n_estimators=50,
estimator=nb_clf, learning_rate=1.0)
adaboost_nb_clf.fit(X_train, y_train)
y_pred_adaboost = adaboost_nb_clf.predict(X_test)

print("Adaboost with Naive Bayes :")
print("Accuracy: ", accuracy_score(y_test,
y_pred_adaboost))
print("Precision: ", precision_score(y_test,
y_pred_adaboost))
print("Recall: ", recall_score(y_test,
y_pred_adaboost))
```

```
print("F1-Score: ", f1_score(y_test,
y_pred_adaboost))
```

File - Day12Q3

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day12Q3.py
Naive Bayes Classifier - Default Model Performance Metrics:
Accuracy:  0.7207792207792207
Precision:  0.6190476190476191
Recall:  0.49056603773584906
F1-Score:  0.5473684210526316
Naive Bayes Classifier - Cross-Validated F1-Score: 0.6418126854702277
Adaboost with Naive Bayes :
Accuracy:  0.6168831168831169
Precision:  0.38461538461538464
Recall:  0.18867924528301888
F1-Score:  0.25316455696202533

Process finished with exit code 0

```