

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())
```

```

1 C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
  tejas\PycharmProjects\pythonProject\START\Day12Q1.py
2 <class 'pandas.core.frame.DataFrame'>
3 RangeIndex: 768 entries, 0 to 767
4 Data columns (total 7 columns):
5 #   Column                Non-Null Count  Dtype
6 ---  ---
7 0   Pregnancies            768 non-null    int64
8 1   Glucose                768 non-null    int64
9 2   BloodPressure          768 non-null    int64
10 3   BMI                   768 non-null    float64
11 4   DiabetesPedigreeFunction 768 non-null    float64
12 5   Age                   768 non-null    int64
13 6   Outcome                768 non-null    int64
14 dtypes: float64(2), int64(5)
15 memory usage: 42.1 KB
16 None
17 Pregnancies            0.0
18 Glucose                0.0
19 BloodPressure          0.0
20 BMI                   0.0
21 DiabetesPedigreeFunction 0.0
22 Age                   0.0
23 Outcome                0.0
24 dtype: float64
25 Pregnancies            0.901674
26 Glucose                0.173754
27 BloodPressure          -1.843608
28 BMI                   -0.428982
29 DiabetesPedigreeFunction 1.919911
30 Age                   1.129597
31 Outcome                0.635017
32 dtype: float64
33 <class 'pandas.core.frame.DataFrame'>
34 RangeIndex: 768 entries, 0 to 767
35 Data columns (total 7 columns):
36 #   Column                Non-Null Count  Dtype
37 ---  ---
38 0   Pregnancies            768 non-null    int64
39 1   Glucose                768 non-null    int64
40 2   BloodPressure          768 non-null    int64
41 3   BMI                   768 non-null    float64
42 4   DiabetesPedigreeFunction 768 non-null    float64
43 5   Age                   768 non-null    int64
44 6   Outcome                768 non-null    int64
45 dtypes: float64(2), int64(5)
46 memory usage: 42.1 KB
47 None
48
49 Process finished with exit code 0

```

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

```

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()

```

```

1 C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
  tejas\PycharmProjects\pythonProject\START\Day12Q2.py
2 <class 'pandas.core.frame.DataFrame'>
3 RangeIndex: 768 entries, 0 to 767
4 Data columns (total 7 columns):
5 #   Column                Non-Null Count  Dtype
6 ---  ---
7 0   Pregnancies            768 non-null   int64
8 1   Glucose                768 non-null   int64
9 2   BloodPressure          768 non-null   int64
10 3   BMI                    768 non-null   float64
11 4   DiabetesPedigreeFunction 768 non-null   float64
12 5   Age                    768 non-null   int64
13 6   Outcome                768 non-null   int64
14 dtypes: float64(2), int64(5)
15 memory usage: 42.1 KB
16 None
17 Pregnancies            0.0
18 Glucose                0.0
19 BloodPressure          0.0
20 BMI                    0.0
21 DiabetesPedigreeFunction 0.0
22 Age                    0.0
23 Outcome                0.0
24 dtype: float64
25 Pregnancies            0.901674
26 Glucose                0.173754
27 BloodPressure          -1.843608
28 BMI                    -0.428982
29 DiabetesPedigreeFunction 1.919911
30 Age                    1.129597
31 Outcome                0.635017
32 dtype: float64
33 <class 'pandas.core.frame.DataFrame'>
34 RangeIndex: 768 entries, 0 to 767
35 Data columns (total 7 columns):
36 #   Column                Non-Null Count  Dtype
37 ---  ---
38 0   Pregnancies            768 non-null   int64
39 1   Glucose                768 non-null   int64
40 2   BloodPressure          768 non-null   int64
41 3   BMI                    768 non-null   float64
42 4   DiabetesPedigreeFunction 768 non-null   float64
43 5   Age                    768 non-null   int64
44 6   Outcome                768 non-null   int64
45 dtypes: float64(2), int64(5)
46 memory usage: 42.1 KB
47 None
48 4
49 <class 'pandas.core.frame.DataFrame'>
50 Index: 764 entries, 0 to 767
51 Data columns (total 7 columns):
52 #   Column                Non-Null Count  Dtype
53 ---  ---
54 0   Pregnancies            764 non-null   int64
55 1   Glucose                764 non-null   int64
56 2   BloodPressure          764 non-null   int64
57 3   BMI                    764 non-null   float64
58 4   DiabetesPedigreeFunction 764 non-null   float64

```

```

59 5   Age                764 non-null   int64
60 6   Outcome            764 non-null   int64
61 dtypes: float64(2), int64(5)
62 memory usage: 47.8 KB
63 4
64 <class 'pandas.core.frame.DataFrame'>
65 Index: 764 entries, 0 to 767
66 Data columns (total 7 columns):
67 #   Column                Non-Null Count  Dtype
68 ---  ---
69 0   Pregnancies            764 non-null    int64
70 1   Glucose                764 non-null    int64
71 2   BloodPressure          764 non-null    int64
72 3   BMI                    764 non-null    float64
73 4   DiabetesPedigreeFunction 764 non-null    float64
74 5   Age                    764 non-null    int64
75 6   Outcome                764 non-null    int64
76 dtypes: float64(2), int64(5)
77 memory usage: 47.8 KB
78 4
79 <class 'pandas.core.frame.DataFrame'>
80 Index: 764 entries, 0 to 767
81 Data columns (total 7 columns):
82 #   Column                Non-Null Count  Dtype
83 ---  ---
84 0   Pregnancies            764 non-null    int64
85 1   Glucose                764 non-null    int64
86 2   BloodPressure          764 non-null    int64
87 3   BMI                    764 non-null    float64
88 4   DiabetesPedigreeFunction 764 non-null    float64
89 5   Age                    764 non-null    int64
90 6   Outcome                764 non-null    int64
91 dtypes: float64(2), int64(5)
92 memory usage: 47.8 KB
93 4
94 <class 'pandas.core.frame.DataFrame'>
95 Index: 764 entries, 0 to 767
96 Data columns (total 7 columns):
97 #   Column                Non-Null Count  Dtype
98 ---  ---
99 0   Pregnancies            764 non-null    int64
100 1   Glucose                764 non-null    int64
101 2   BloodPressure          764 non-null    int64
102 3   BMI                    764 non-null    float64
103 4   DiabetesPedigreeFunction 764 non-null    float64
104 5   Age                    764 non-null    int64
105 6   Outcome                764 non-null    int64
106 dtypes: float64(2), int64(5)
107 memory usage: 47.8 KB
108 4
109 <class 'pandas.core.frame.DataFrame'>
110 Index: 764 entries, 0 to 767
111 Data columns (total 7 columns):
112 #   Column                Non-Null Count  Dtype
113 ---  ---
114 0   Pregnancies            764 non-null    int64
115 1   Glucose                764 non-null    int64
116 2   BloodPressure          764 non-null    int64
117 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'.

- Use Naïve Bayes Classifier algorithm to classify *Outcome* and print the default model performance metrics: Accuracy, Precision, Recall, F1Score.
- Use k-Fold Cross Validation to find the best model performance metrics .
- 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

```
1 C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\  
tejas\PycharmProjects\pythonProject\START\Day12Q3.py  
2 Naive Bayes Classifier - Default Model Performance Metrics:  
3 Accuracy: 0.7207792207792207  
4 Precision: 0.6190476190476191  
5 Recall: 0.49056603773584906  
6 F1-Score: 0.5473684210526316  
7 Naive Bayes Classifier - Cross-Validated F1-Score: 0.6418126854702277  
8 Adaboost with Naive Bayes :  
9 Accuracy: 0.6168831168831169  
10 Precision: 0.38461538461538464  
11 Recall: 0.18867924528301888  
12 F1-Score: 0.25316455696202533  
13  
14 Process finished with exit code 0  
15
```