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())
```

File - Day9Q1

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day9Q1.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.

```
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()
```

File - Day9Q2

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day9Q2.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
```

```
 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 - Day9Q2

```
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 75% training and 25% testing data. Then, use a SVM classifier algorithm with target variable as 'Outcome'.

a. Print the default model performance metrics: Accuracy, Precision, Recall, F1Score

b. Plot a Precision & Recall & F1-Score vs kernel type(**'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'**) curve (All metrics on the same graph). Find the kernel type for which F1-score is the highest.

c. Plot a curve on Precision & Recall & F1-Score vs **appropriate range of C** using the best kernel type you obtained in question(3b), (All metrics on the same graph). Find the C for which F1-score is the highest for the given kernel type.

(Since, it was taking a ton of time to load, with due permission from Akash sir, I have only inserted the snapshot till performance metrices.)

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import
train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score,
precision_score, recall_score, f1_score

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)]

df_OutlierFree = df.drop(OutlierData.index, axis=0)
df_OutlierFree.info()

X = df_OutlierFree.drop('Outcome', axis=1)  # all
columns except 'Outcome'
y = df_OutlierFree['Outcome']  # target Variable
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.25,
random_state=1234)
unique_classes = np.unique(y)
print('Number of unique classes:',
len(unique_classes))
```

```
kernel_values = ['linear', 'poly', 'rbf',
'sigmoid']

#  Create a svm Classifier
svm_clf = svm.SVC(kernel=kernel_values[0], C=1,
gamma=0.1)  # Linear Kernel

# Train the model using the training sets
svm_clf = svm_clf.fit(X_train, y_train)

# Predict the response for test dataset
y_pred = svm_clf.predict(X_test)

print("Model Performance metrics are as below :-
\n")
print("Accuracy is " + str(accuracy_score(y_test,
y_pred)))
print("Precision is " + str(precision_score(y_test,
y_pred, zero_division=1)))
print("Recall is " + str(recall_score(y_test,
y_pred)))
print("F1-Score is " + str(f1_score(y_test,
y_pred)))

# C_values = [0.001,0.01,0.1,1,10,100,1000] # C of
1 is a good starting point || C > 0
C_values = [0.01, 0.1, 1, 10]
# gamma_values = [0.1,0.5,0.9] || Gamma > 0 ||
starting value : gamma = 1/len(X) = 1/no. of rows
in the dataset
gamma_values = [0.1, 1 / len(X)]
kernel_values = ['linear', 'poly', 'rbf',
'sigmoid']

PerfData = pd.DataFrame(columns=['Kernel Type',
'C', 'Gamma', 'Precision', 'Recall', 'Accuracy',
'F1-Score'])

for k in kernel_values:
    for c in C_values:
        for g in gamma_values:
            svm_clf = svm.SVC(kernel=k, C=c,
```

```
gamma=g)
            svm_clf = svm_clf.fit(X_train, y_train)
            y_pred = svm_clf.predict(X_test)

            row = [
                [k, c, g, precision_score(y_test,
y_pred, zero_division=1), recall_score(y_test,
y_pred), accuracy_score(y_test, y_pred),
                 f1_score(y_test, y_pred)]]
            df2 = pd.DataFrame(row,
columns=['Kernel Type', 'C', 'Gamma', 'Precision',
'Recall', 'Accuracy', 'F1-Score'])
            PerfData = pd.concat([PerfData, df2],
ignore_index=True)

print(PerfData.head(10))
print(PerfData[PerfData['F1-Score'] ==
max(PerfData['F1-Score'])])
plt.plot(kernel_values, PerfData[PerfData['Kernel
Type'] == 'linear']['Precision'],
label='Precision')
plt.plot(kernel_values, PerfData[PerfData['Kernel
Type'] == 'linear']['Recall'], label='Recall')
plt.scatter(kernel_values,
PerfData[PerfData['Kernel Type'] == 'linear']['F1-
Score'], label='F1-Score')
plt.xlabel('Kernel type')
plt.ylabel('Score')
plt.title('Precision & Recall & F-1 Score vs kernel
type')
plt.legend(loc='upper right')
plt.show()
svm_best_kernel = SVC(kernel=best_kernel)
precisions = []
recalls = []
f1_scores = []
C_values = np.arange(0.1, 10, 0.1)
for C in C_values:
    svm_best_kernel.set_params(C=C)
    svm_best_kernel.fit(X_train, y_train)
    y_pred = svm_best_kernel.predict(X_test)
    precision = precision_score(y_test, y_pred)
```

```
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1)

best_C = C_values[np.argmax(f1_scores)]
print(f"The C value with the highest F1-score for
kernel type '{best_kernel}' is: {best_C}")

plt.plot(C_values, precisions, label='Precision')
plt.plot(C_values, recalls, label='Recall')
plt.plot(C_values, f1_scores, label='F1-Score')
plt.xlabel('C')
plt.ylabel('Score')
plt.title(f"Precision, Recall, and F1-Score vs C
for Kernel Type '{best_kernel}'")
plt.legend()
plt.show()
```

```
  tejas\PycharmProjects\pythonProject\START\Day9Q3.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
<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
```

File - Day9Q3

```
 6   Outcome                   764 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 47.8 KB
Number of unique classes: 2
Model Performance metrics are as below :-

Accuracy is 0.7382198952879581
Precision is 0.717391304347826
Recall is 0.4714285714285714
F1-Score is 0.5689655172413792

```