## 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"].
mean(), 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\Day10Q1.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
                               768 non-null int64
      Glucose
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
50
```

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

```
df = pd.read csv('Dataset Day7.csv')
print(df.info())
missing value percent = df.isna().sum() / len(df) *
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] <</pre>
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\Day10Q2.py
 2 <class 'pandas.core.frame.DataFrame'>
 3 RangeIndex: 768 entries, 0 to 767
 4 Data columns (total 7 columns):
       Column
                                 Non-Null Count Dtype
 6 ---
       -----
                                 -----
 7
   Θ
       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
       DiabetesPedigreeFunction 768 non-null
11 4
                                                float64
12 5
       Age
                                768 non-null
                                                int64
13 6
                                768 non-null
      Outcome
                                                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
                                 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, θ to 767
51 Data columns (total 7 columns):
52 # Column
                                Non-Null Count Dtype
53 ---
      Pregnancies
54 0
                                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
```

```
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
                                764 non-null int64
 70 1
        Glucose
 71 2
        BloodPressure
                                764 non-null int64
 72 3
                                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):
                                Non-Null Count Dtype
 82 # Column
 83 ---
        -----
                                -----
 84 0 Pregnancies
                                764 non-null
                                              int64
                                              int64
 85 1
                                764 non-null
        Glucose
                                              int64
 86 2
                                764 non-null
        BloodPressure
 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
 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
                                764 non-null float64
102 3
       BMI
103 4
        DiabetesPedigreeFunction 764 non-null float64
104 5
                                764 non-null
       Age
                                               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
116 2
                                764 non-null
        Glucose
                                               int64
        BloodPressure
                                764 non-null
                                              int64
                                744 non-null
                                              £100±44
```

## File - Day10Q2

```
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
124 Process finished with exit code 0
125
```

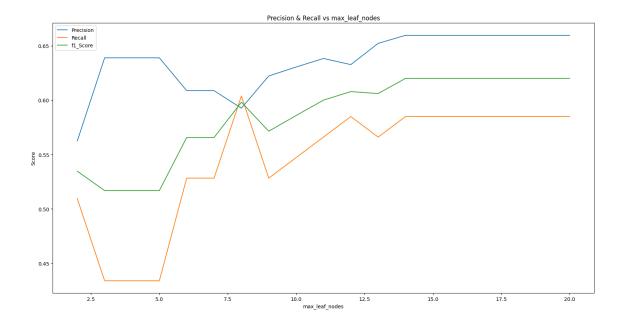
- 3. Split the data into 80% training and 20% testing data. Then, use a Decision Tree classifier algorithm with target variable as 'Outcome'.
  - a. Print the default model performance metrics: Accuracy, Precision, Recall, F1Score

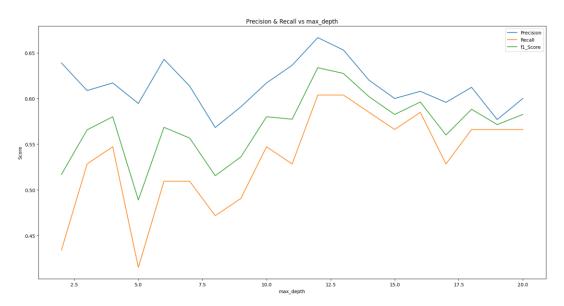
- b. Plot a Precision & Recall vs max\_leaf\_nodes (consider a range of numbers) curve (both Prec and Rec on the same graph). Find the kernel type for which F1-score is the highest.
- c. Plot a Precision & Recall vs max\_depth (**consider a range of numbers**) curve (both Prec and Rec on the same graph). Find the kernel type for which F1-score is the highest.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model selection import
train test split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy score,
precision score, recall score, f1 score
# Load the dataset
df = pd.read csv('Dataset Day7.csv')
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)
X = df.drop('Outcome', axis=1)  # Features
y = df['Outcome'] # Target variable
X train, X test, y train, y test =
train test split(X, y, test size=0.2,
# Part (a) - Default model performance metrics
dt clf = DecisionTreeClassifier()
dt clf.fit(X train, y train)
y pred = dt clf.predict(X test)
print("Model Performance metrics (Default
```

```
Parameters):\n")
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))
# Part (b) - Plot Precision & Recall vs
max leaf nodes
max leaf nodes = np.arange(2, 21)
precisions = []
recalls = []
f1 scores = []
for nodes in max leaf nodes:
    dt clf =
DecisionTreeClassifier(max leaf nodes=nodes)
    dt clf.fit(X train, y train)
    y pred = dt clf.predict(X test)
    precisions.append(precision score(y test,
y pred))
    recalls.append(recall score(y test, y pred))
    f1 scores.append(f1 score(y test, y pred))
best max leaf node =
max leaf nodes[np.argmax(f1 scores)]
print("Best max leaf nodes for highest F1-Score:",
best max leaf node)
plt.plot(max leaf nodes, precisions,
label='Precision')
plt.plot(max leaf nodes, recalls, label='Recall')
plt.plot(max leaf nodes, f1 scores,
plt.xlabel('max leaf nodes')
plt.ylabel('Score')
plt.title('Precision & Recall vs max leaf nodes')
plt.legend()
plt.show()
# Part (c) - Plot Precision & Recall vs max depth
max depths = np.arange(2, 21)
precisions = []
```

```
recalls = []
f1 \text{ scores} = []
for depth in max depths:
    dt clf =
DecisionTreeClassifier(max depth=depth)
    dt clf.fit(X train, y train)
    y pred = dt clf.predict(X test)
    precisions.append(precision score(y test,
y pred))
    recalls.append(recall score(y test, y pred))
    f1 scores.append(f1 score(y test, y pred))
best max depth = max depths[np.argmax(f1 scores)]
print("Best max depth for highest F1-Score:",
best max depth)
plt.plot(max depths, precisions, label='Precision')
plt.plot(max_depths, recalls, label='Recall')
plt.plot(max depths, f1 scores, label='f1 Score')
plt.xlabel('max depth')
plt.ylabel('Score')
plt.title('Precision & Recall vs max depth')
plt.legend()
plt.show()
```





```
1 C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day10Q3.py
2 Model Performance metrics (Default Parameters):
3
4 Accuracy: 0.72727272727273
5 Precision: 0.6122448979591837
6 Recall: 0.5660377358490566
7 F1-Score: 0.588235294117647
8 Best max leaf nodes for highest F1-Score: 14
9 Best max depth for highest F1-Score: 12
10
11 Process finished with exit code 0
12
```