

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())
```

File - Day10Q1

```
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   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
50
```

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\Day10Q2.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 - Day10Q2

```
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. 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')

# Handle missing values
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)

# Split the data into training and testing sets
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,
random_state=50)

# 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,
label='f1_Score')
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 = []

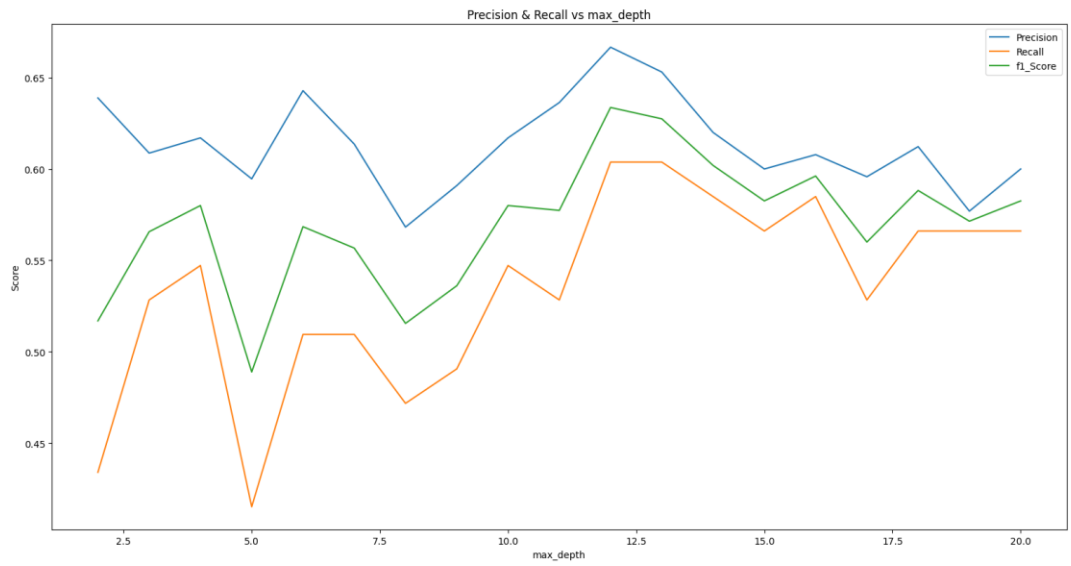
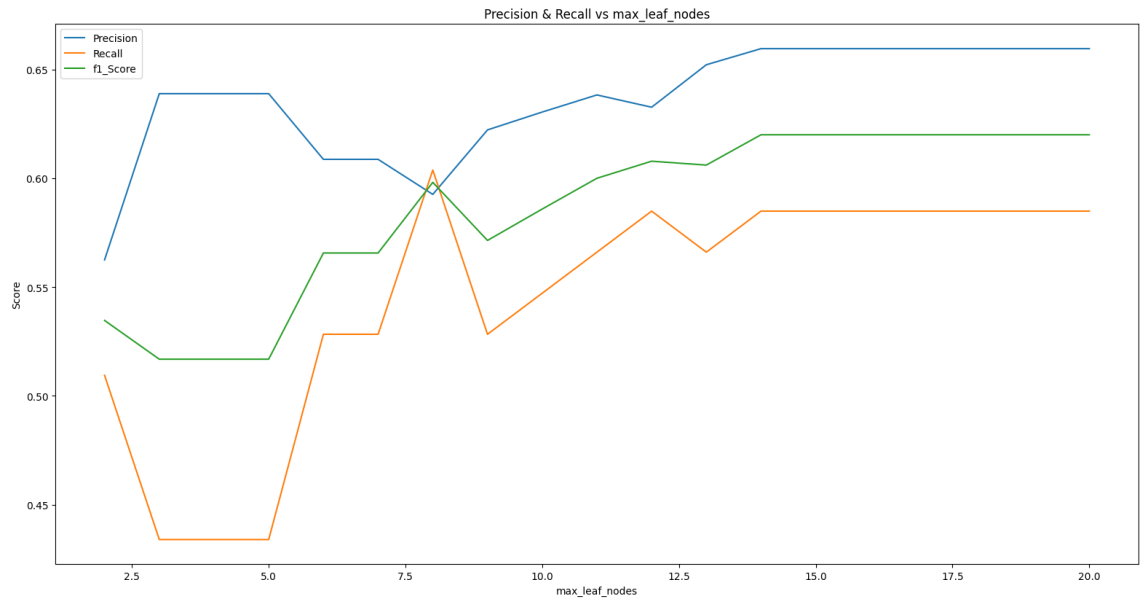
```



```
recalls = []
f1_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()
```



File - Day10Q3

```
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:  0.7272727272727273
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
```