P1

```
import numpy as np, pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("uciml/pima-indians-diabetes-database")

print("Path to dataset files:", path)
```

```
Path to dataset files: /kaggle/input/pima-indians-diabetes-database
```

```
df= pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.csv')
```

```
print("Shape of the dataset:", df.shape)
```

```
Shape of the dataset: (768, 9)
```

```
print("Dataset information:")
print(df.info())
```

```
Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 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   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
```

```
print(df.describe())
```

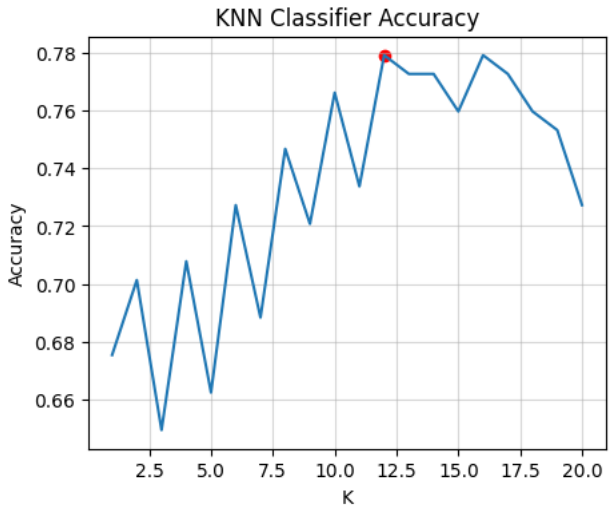
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000   
mean      3.845052  120.894531      69.105469      20.536458   79.799479   
std       3.369578   31.972618      19.355807      15.952218  115.244002   
min       0.000000    0.000000       0.000000       0.000000    0.000000   
25%       1.000000   99.000000      62.000000       0.000000    0.000000   
50%       3.000000  117.000000      72.000000      23.000000   30.500000   
75%       6.000000  140.250000      80.000000      32.000000  127.250000   
max      17.000000  199.000000     122.000000      99.000000  846.000000   

              BMI  DiabetesPedigreeFunction         Age     Outcome  
count  768.000000                768.000000  768.000000  768.000000  
mean    31.992578                  0.471876   33.240885    0.348958  
std      7.884160                  0.331329   11.760232    0.476951  
min      0.000000                  0.078000   21.000000    0.000000  
25%     27.300000                  0.243750   24.000000    0.000000  
50%     32.000000                  0.372500   29.000000    0.000000  
75%     36.600000                  0.626250   41.000000    1.000000  
max     67.100000                  2.420000   81.000000    1.000000  
```

```
from sklearn.model_selection import train_test_split

X,y=df.drop("Outcome",axis=1),df["Outcome"]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)
```

```
(614, 8) (154, 8) (614,) (154,)
```

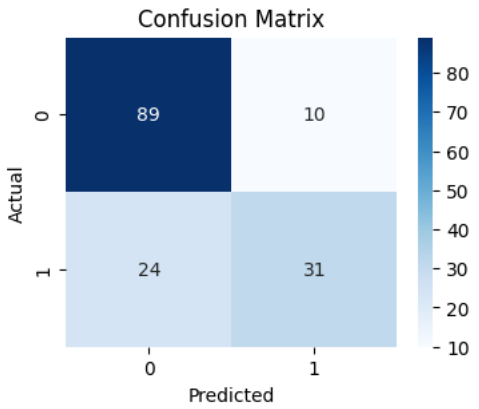
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
acc=[]
bk,ba=1,0

for k in range(1,21):
  knn=KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train,y_train)

  cm = confusion_matrix(y_test,knn.predict(X_test))
  tp, tn, fp, fn = cm.ravel()
  acc_val, rec, prec = (tp+tn)/(tp+tn+fp+fn), tp/(tp+fn), tp/(tp+fp)
  f1 = 2*rec*prec/(rec+prec)
  print(f"For k={k} -->\tAccuracy: {acc_val:.4f}, Recall: {rec:.4f}, Precision: {prec:.4f}, F1 Score: {f1:.4f}")

  acc.append(knn.score(X_test,y_test))
  if knn.score(X_test,y_test)>ba:
    ba=knn.score(X_test,y_test);bk=k

print(f"\n\nBest k: {bk}, Accuracy: {ba:.4f}\n")

plt.figure(figsize=(5,4))
plt.scatter(bk,ba,color="red")
plt.plot(range(1,21),acc)
plt.xlabel("K")
plt.ylabel("Accuracy")
plt.title("KNN Classifier Accuracy")
plt.grid(alpha=0.5)
plt.show()
```

```
For k=1 -->     Accuracy: 0.6429, Recall: 0.6731, Precision: 0.7692, F1 Score: 0.7179
For k=2 -->     Accuracy: 0.6429, Recall: 0.8148, Precision: 0.7154, F1 Score: 0.7619
For k=3 -->     Accuracy: 0.6429, Recall: 0.6500, Precision: 0.7647, F1 Score: 0.7027
For k=4 -->     Accuracy: 0.6429, Recall: 0.7523, Precision: 0.7455, F1 Score: 0.7489
For k=5 -->     Accuracy: 0.6429, Recall: 0.6863, Precision: 0.7527, F1 Score: 0.7179
For k=6 -->     Accuracy: 0.6429, Recall: 0.7232, Precision: 0.7714, F1 Score: 0.7465
For k=7 -->     Accuracy: 0.6429, Recall: 0.6792, Precision: 0.7742, F1 Score: 0.7236
For k=8 -->     Accuracy: 0.6429, Recall: 0.7043, Precision: 0.7941, F1 Score: 0.7465
For k=9 -->     Accuracy: 0.6429, Recall: 0.6667, Precision: 0.8043, F1 Score: 0.7291
For k=10 -->    Accuracy: 0.6429, Recall: 0.7288, Precision: 0.7890, F1 Score: 0.7577
For k=11 -->    Accuracy: 0.6429, Recall: 0.6991, Precision: 0.7900, F1 Score: 0.7418
For k=12 -->    Accuracy: 0.6429, Recall: 0.7417, Precision: 0.7876, F1 Score: 0.7639
For k=13 -->    Accuracy: 0.6429, Recall: 0.7143, Precision: 0.8019, F1 Score: 0.7556
For k=14 -->    Accuracy: 0.6429, Recall: 0.7479, Precision: 0.7807, F1 Score: 0.7639
For k=15 -->    Accuracy: 0.6429, Recall: 0.7179, Precision: 0.7925, F1 Score: 0.7534
For k=16 -->    Accuracy: 0.6429, Recall: 0.7500, Precision: 0.7826, F1 Score: 0.7660
For k=17 -->    Accuracy: 0.6429, Recall: 0.7311, Precision: 0.7909, F1 Score: 0.7598
For k=18 -->    Accuracy: 0.6429, Recall: 0.7607, Precision: 0.7672, F1 Score: 0.7639
For k=19 -->    Accuracy: 0.6429, Recall: 0.7328, Precision: 0.7798, F1 Score: 0.7556
For k=20 -->    Accuracy: 0.6429, Recall: 0.7679, Precision: 0.7478, F1 Score: 0.7577


Best k: 12, Accuracy: 0.7792
```




```python
  knn=KNeighborsClassifier(n_neighbors=bk)
  knn.fit(X_train,y_train)

  cm = confusion_matrix(y_test,knn.predict(X_test))
  plt.figure(figsize=(4,3))
  sns.heatmap(cm,annot=True,fmt="d",cmap="Blues")
  plt.xlabel("Predicted")
```

```
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, knn.predict(X_test))
print("\n\nMSE:", mse)
```




```
MSE: 0.22077922077922077
```

P2

```
import numpy as np, pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
data=pd.DataFrame(
    np.array([
        ['A',160,50,0],
        ['B',165,55,0],
        ['C',170,65,1],
        ['D',175,70,1],
        ['E',180,80,1]
    ]),
    columns=['ID','Height(cm)','Weight(kg)','Class']
)

print(data)
```

```
  ID Height(cm) Weight(kg) Class
0  A        160         50     0
1  B        165         55     0
2  C        170         65     1
3  D        175         70     1
4  E        180         80     1
```

```
def knn_predict(X, y, x, k):
  eucli_dist = np.sqrt(np.sum((X - x)**2, axis=1))
  sorted_dist_idx = np.argsort(eucli_dist)
  k_nearest_labels = y[sorted_dist_idx[:k]]
  unique_labels, label_counts = np.unique(k_nearest_labels, return_counts=True)
  predicted_label = unique_labels[np.argmax(label_counts)]
  return predicted_label

data['Height(cm)'] = pd.to_numeric(data['Height(cm)'])
data['Weight(kg)'] = pd.to_numeric(data['Weight(kg)'])

for k in range(1,5):
  print(f"For k={k}, Prediction --> {knn_predict(data[['Height(cm)','Weight(kg)']].values, data['Class'].values, np.array(
```

```
For k=1, Prediction --> 1
For k=2, Prediction --> 1
For k=3, Prediction --> 1
For k=4, Prediction --> 1
```