Q1)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

df=pd.read_csv("diabetes.csv.")

df=df.dropna()

df.isnull().sum()

X = df.drop("Outcome", axis=1)
y = df["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

accuracy_scores = []
```

```
values=range(1,21)
for k in values:
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test, y_pred)
accuracy_scores.append(acc)



optimal_k = np.argmax(accuracy_scores) + 1
print("\nOptimal value of K:" ,optimal_k)

plt.plot(values, accuracy_scores)
plt.title("KNN - Accuracy vs K Value")
plt.xlabel("Number of Neighbors (K)")
plt.ylabel("Accuracy")
plt.show()

ypred = knn.predict(X_test)
accuracy= accuracy_score(y_test, y_pred)
print('Accuracy of the model with optimal K=',accuracy)

newpatient = np.array([[6, 148, 72, 35, 0, 33.6, 0.627, 50]])
new_patient_scaled = scaler.transform(newpatient)
prediction = knn.predict(new_patient_scaled)
```

```
if prediction[0] == 1:
print("The new patient is diabetic.")
else:
print("The new patient is non-diabetic.")
```

Q2)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

data=pd.read_csv("BankNote_Authentication.csv")
print(data)

df=df.dropna()

df.isnull().sum()

X = data.drop('class', axis=1)
y = data['class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
model = DecisionTreeClassifier(random_state=50)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```