

## 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  
from sklearn.metrics import accuracy_score, confusion_matrix,  
classification_report
```

```
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,  
random_state=42)
```

```
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 numpy as np
```

```
def relu(x):  
    return np.maximum(0,x)  
  
def sigmoid(x):  
    return 1/(1 + np.exp(-x))
```

```
W1 = np.random.randn(3,4)
```

```
b1 = np.random.randn(4)
```

```
W2 = np.random.randn(4,3)
```

```
b2 = np.random.randn(3)
```

```
W3 = np.random.randn(3, 1)
```

```
b3 = np.random.randn(1)
```

```
def forward(X):
    h1 = relu(np.dot(X, W1) + b1) # 1st hidden layer
    h2 = relu(np.dot(h1, W2) + b2) # 2nd hidden layer
    out = sigmoid(np.dot(h2, W3) + b3) # Output layer
    return out
```

```
X = np.array([[0.1, 0.2, 0.3]])
print("Output:", forward(X))
```