QUESTION 1

Import Libraries, Load and Prepare Data

Scale the Feature, Perform Grid Search,Evaluate Best Model

```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import numpy as np

# Step 1: Load MNIST
mnist = fetch_openml('mnist_784', version=1, as_frame=False)
X, y = mnist['data'], mnist['target'].astype(np.uint8)

# Step 2: Split into train/test
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]

# Step 3: Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 4: Grid Search (no PCA now)
param_grid = {
    'n_neighbors': [3, 4, 5, 6, 7],
    'weights': ['uniform', 'distance']
}
knn_clf = KNeighborsClassifier()

grid_search = GridSearchCV(knn_clf, param_grid, cv=3, n_jobs=-1,
verbose=1)
grid_search.fit(X_train_scaled, y_train)

# Step 5: Evaluate
best_knn = grid_search.best_estimator_
y_pred = best_knn.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
```

```
print("Best Parameters:", grid_search.best_params_)
print("Test Set Accuracy:", round(accuracy, 4))
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/datasets/
_openml.py:968: FutureWarning: The default value of `parser` will
change from `'liac-arff'` to `'auto'` in 1.4. You can set
`parser='auto'` to silence this warning. Therefore, an `ImportError`
will be raised from 1.4 if the dataset is dense and pandas is not
installed. Note that the pandas parser may return different data
types. See the Notes Section in fetch_openml's API doc for details.
  warn(

Fitting 3 folds for each of 10 candidates, totalling 30 fits
Best Parameters: {'n_neighbors': 4, 'weights': 'distance'}
Test Set Accuracy: 0.9489
```

QUESTION 2

```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import numpy as np

# Step 1: Load the MNIST dataset
mnist = fetch_openml('mnist_784', version=1, as_frame=False)
X, y = mnist['data'], mnist['target'].astype(np.uint8)

# Step 2: Split the dataset
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]

# Step 3: Define image shifting function
def shift_image(image, direction):
    image = image.reshape(28, 28)
    shifted = np.roll(image, shift=1, axis=direction)  # 0=down,
1=right
    if direction == 0:  # down
        shifted[0, :] = 0
    else:  # right
        shifted[:, 0] = 0
    return shifted.reshape(784)

# Step 4: Create augmented dataset (original + shifted versions)
X_augmented = [X_train]
y_augmented = [y_train]
```

```
for direction in [0, 1, -1, -2]:  # 0=down, 1=right, -1=up, -2=left
    X_shifted = np.apply_along_axis(shift_image, 1, X_train, direction
% 2)
    if direction < 0:  # if shifting up or left, roll in opposite
direction
        X_shifted = np.roll(X_shifted.reshape(-1, 28, 28), shift=-1,
axis=direction % 2).reshape(-1, 784)
        if direction == -1:
            X_shifted[-1, :] = 0
        else:
            X_shifted[:, -1] = 0
    X_augmented.append(X_shifted)
    y_augmented.append(y_train)

X_train_augmented = np.concatenate(X_augmented)
y_train_augmented = np.concatenate(y_augmented)

# Step 5: Scale the data
scaler = StandardScaler()
X_train_aug_scaled = scaler.fit_transform(X_train_augmented)
X_test_scaled = scaler.transform(X_test)

# Step 6: Train the model
knn_clf = KNeighborsClassifier(n_neighbors=4, weights='distance')
knn_clf.fit(X_train_aug_scaled, y_train_augmented)

# Step 7: Evaluate on test set
y_pred = knn_clf.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)

# Step 8: Print the results
print("Test Accuracy after augmentation:", round(accuracy, 4))
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/datasets/
_openml.py:968: FutureWarning: The default value of `parser` will
change from `'liac-arff'` to `'auto'` in 1.4. You can set
`parser='auto'` to silence this warning. Therefore, an `ImportError`
will be raised from 1.4 if the dataset is dense and pandas is not
installed. Note that the pandas parser may return different data
types. See the Notes Section in fetch_openml's API doc for details.
  warn(

Test Accuracy after augmentation: 0.9536
```