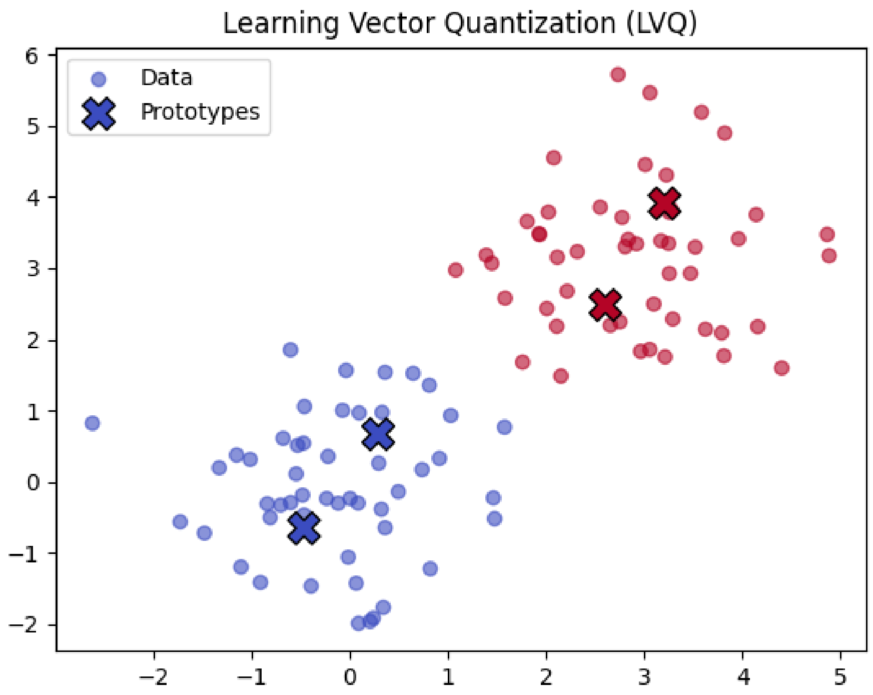
```
import numpy as np
import matplotlib.pyplot as plt

class LVQ:
  def __init__(self, learning_rate=0.1,
epochs=100):
    self.learning_rate = learning_rate
    self.epochs = epochs
    self.prototypes = None
    self.prototype_labels = None

  def fit(self, X, y, n_prototypes_per_class=1):
    classes = np.unique(y)
    n_features = X.shape[1]
    self.prototypes = []
    self.prototype_labels = []
    for cls in classes:
      idx = np.where(y == cls)[0]
      chosen = X[np.random.choice(idx,
n_prototypes_per_class, replace=False)]
      self.prototypes.extend(chosen)
      self.prototype_labels.extend([cls] *
n_prototypes_per_class)
    self.prototypes = np.array(self.prototypes)
    self.prototype_labels =
np.array(self.prototype_labels)
    for epoch in range(self.epochs):
      for i, x in enumerate(X):
        label = y[i]
        distances =
np.linalg.norm(self.prototypes - x, axis=1)
        winner_idx = np.argmin(distances)
        if self.prototype_labels[winner_idx] ==
label:
          self.prototypes[winner_idx] +=
self.learning_rate * (x -
self.prototypes[winner_idx])
        else:
          self.prototypes[winner_idx] -=
self.learning_rate * (x -
self.prototypes[winner_idx])
      self.learning_rate *= 0.95


  def predict(self, X):
    preds = []
    for x in X:
      distances = np.linalg.norm(self.prototypes
- x, axis=1)
      winner_idx = np.argmin(distances)
      preds.append(self.prototype_labels[winne
r_idx])
    return np.array(preds)

if __name__ == "__main__":
  np.random.seed(42)
  class0 = np.random.randn(50, 2) + np.array([0,
0])
  class1 = np.random.randn(50, 2) + np.array([3,
3])
  X = np.vstack((class0, class1))
  y = np.array([0]*50 + [1]*50)
  lvq = LVQ(learning_rate=0.3, epochs=20)
  lvq.fit(X, y, n_prototypes_per_class=2)
  preds = lvq.predict(X)
  acc = np.mean(preds == y)
  print(f"Training Accuracy: {acc:.2f}")
  plt.scatter(X[:,0], X[:,1], c=y,
cmap="coolwarm", alpha=0.6, label="Data")
  plt.scatter(lvq.prototypes[:,0],
lvq.prototypes[:,1],
        c=lvq.prototype_labels,
cmap="coolwarm", edgecolors="k", marker="X",
s=200, label="Prototypes")
  plt.title("Learning Vector Quantization (LVQ)")
  plt.legend()
  plt.show()
```

OUTPUT

```
Training Accuracy: 1.00
```


Learning Vector Quantization (LVQ)
Data
Prototypes