Problem 2

```
import numpy as np
from scipy.optimize import minimize
from sklearn.svm import SVC

# ---------------------------------------------------------------------
# (A) SciPy-based Hard-Margin SVM
# ---------------------------------------------------------------------
print("========== PROBLEM 2 (Part 2) - SciPy Version ==========\n")

# 1) Load the data
X = np.genfromtxt("nci.data.csv", delimiter=",", dtype=float).T
X = np.nan_to_num(X, nan=0.0)  # fill any NaNs if present

with open("nci.label.txt", "r") as f:
    labels = [line.strip() for line in f if line.strip() != ""]

# If there's a mismatch in lengths, patch with a dummy label
if len(labels) < X.shape[0]:
    labels.append("UNKNOWN")

labels = np.array(labels)

# Filter for NSCLC (+1) vs RENAL (-1)
mask = (labels == "NSCLC") | (labels == "RENAL")
```

```
X = X[mask]
y = np.where(labels[mask] == "NSCLC", 1, -1)

# 2) Construct the SVM Dual
K = X @ X.T
P = np.outer(y, y) * K
n = X.shape[0]

def svm_dual_objective(alpha):
    return 0.5 * alpha @ P @ alpha - np.sum(alpha)

def svm_dual_constraint(alpha):
    return np.dot(alpha, y)

bounds = [(0, None)] * n
constraints = {"type": "eq", "fun": svm_dual_constraint}
alpha0 = np.zeros(n)

# 3) Solve the constrained QP with SciPy
res = minimize(svm_dual_objective, alpha0, bounds=bounds, constraints=constraints)
alphas = res.x

# (a) Dual variables
print("(a) Dual variables (alphas):\n", alphas, "\n")

# (b) Slope vector w
```

```
w = ((alphas * y)[:, None] * X).sum(axis=0)
print("(b) Slope vector (w):\n", w, "\n")

# (c) Intercept b
sv_indices = np.where(alphas > 1e-5)[0]
b = np.mean([y[i] - np.dot(w, X[i]) for i in sv_indices])
print("(c) Intercept (b):\n", b, "\n")

# (d) Support vector indices
print("(d) Support vector indices:\n", sv_indices, "\n")

# (e) Distances & Accuracy
distances = (X @ w + b) / np.linalg.norm(w)
predictions = np.sign(X @ w + b)
accuracy = np.mean(predictions == y)
print("(e) Distances from hyperplane:\n", distances, "\n")
print("    Accuracy:", accuracy, "\n")

# (f) Margin
margin = 1.0 / np.linalg.norm(w)
print("(f) Margin:", margin, "\n")

# (g) Predict for x=all ones
x_new = np.ones(X.shape[1])
pred_value = np.dot(w, x_new) + b
pred_label = 1 if pred_value >= 0 else -1
```

```
print("(g) Prediction for x=all ones:", "NSCLC" if pred_label == 1 else "RENAL")


# ---------------------------------------------------------------------
# (B) Scikit-learn-based Hard-Margin SVM
# ---------------------------------------------------------------------
print("\n========== PROBLEM 2 (Part 2) - Scikit-Learn Version ==========\n")

clf = SVC(kernel="linear", C=1e10)  # large C => approximate hard margin
clf.fit(X, y)

# (a) Dual variables:
# scikit-learn only stores alpha_i*y_i for support vectors in clf.dual_coef_[0].
# We can reconstruct the full alpha array, with zeros for non-SVs:
n = len(y)
alphas_skl = np.zeros(n, dtype=float)
alpha_y = clf.dual_coef_[0]       # shape: (1, n_SV), alpha_i * y_i
support_vecs = clf.support_       # indices of support vectors
for i, sv_ix in enumerate(support_vecs):
    alphas_skl[sv_ix] = alpha_y[i] / y[sv_ix]

print("(a) Dual variables (alphas) from scikit-learn:\n", alphas_skl, "\n")

# (b) w
w_skl = clf.coef_.ravel()
print("(b) Slope vector (w):\n", w_skl, "\n")
```

```
# (c) b
b_skl = clf.intercept_[0]
print("(c) Intercept (b):\n", b_skl, "\n")

# (d) Support vector indices
print("(d) Support vector indices:\n", support_vecs, "\n")

# (e) Distances & Accuracy
distances_skl = (X @ w_skl + b_skl) / np.linalg.norm(w_skl)
predictions_skl = clf.predict(X)
accuracy_skl = np.mean(predictions_skl == y)
print("(e) Distances from hyperplane:\n", distances_skl, "\n")
print("    Accuracy:", accuracy_skl, "\n")

# (f) Margin
margin_skl = 1.0 / np.linalg.norm(w_skl)
print("(f) Margin:", margin_skl, "\n")

# (g) Predict for x=all ones
pred_value_skl = np.dot(w_skl, x_new) + b_skl
pred_label_skl = 1 if pred_value_skl >= 0 else -1
print("(g) Prediction for x=all ones:", "NSCLC" if pred_label_skl == 1 else "RENAL")
```

Problem 2 Part 1 SciPy

a. **Dual variables:**

Dual variables (alphas):

[3.70278657e-04 3.01033070e-04 4.46397864e-04 2.49846667e-04

7.01584683e-05 2.55017977e-05 4.46448252e-04 1.50994606e-04

1.14033099e-04 1.74384569e-04 1.68914487e-04 2.48013812e-04

1.78839389e-04 2.33601040e-04 3.42191464e-04 3.73159695e-05

1.15864802e-04 2.55018435e-05]

b. Slope parameter vector

Slope vector (w):

[0.00000000e+00 -5.21888794e-04 7.06950790e-04 ... 4.28827108e-04

-8.30583720e-05 5.08519002e-05]

c. Intercept parameter

Intercept (b):

0.443292031914553

d. Indices of support vectors

Support vector indices:

[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17]

e. Distance of each data point from decision hyperplane and accuracy of classifier

Distances from hyperplane:

[-16.28259853 16.5007701 16.58102246 -16.08756754 -16.56092696

-18.09147119 -16.87225366 -17.2181869 -16.72237254 -17.296374

15.76299432 -16.32106027 16.20170944 15.9985837 16.16599994

16.29085067 16.17330577 21.77757519]

Accuracy: 1.0

f. Margin

Margin: 16.39058040244113

g. Predict the label for all features being 1

Prediction for x=all ones: NSCLC

Problem 2 Part 2 Scikit-learn

a. Dual variables

Dual variables (alphas) from scikit-learn:

[3.75008740e-04 2.97496552e-04 4.43180033e-04 2.56576770e-04

8.30421818e-05 0.00000000e+00 4.50693399e-04 1.35940064e-04

1.24360756e-04 1.64586572e-04 1.77401300e-04 2.57830962e-04

1.85696746e-04 2.40694456e-04 3.44975992e-04 4.06377437e-05

1.17956624e-04 0.00000000e+00]

b. Slope vector w

(w): [0.00000000e+00 -5.47791640e-04 7.03930605e-04 ... 4.14137079e-04

-7.69156096e-05 -2.97353874e-05]

c. Intercept parameter

Intercept (b):

0.4451013446694967

d. Indices of the support vectors

Support vector indices: [0 3 4 6 7 8 9 11 1 2 10 12 13 14 15 16]

e. Distance from decision hyperplane and accuracy of classifier

Distances from hyperplane:

[-16.45109703 16.45406943 16.44994202 -16.44787892 -16.45073654

-16.82901754 -16.44566721 -16.44501055 -16.45398984 -16.44508006

16.45394685 -16.45112354 16.45336919 16.4394348 16.45282843

16.4419348 16.44505715 20.3731282]

Accuracy: 1.0

f. Margin

Margin: 16.44796252351478

g. Prediction for all features = 1

Prediction for x=all ones: NSCLC

Problem 3 Part 1 SciPy

```
import numpy as np
from scipy.optimize import minimize
from sklearn.svm import SVC

# ============================================================
#   PROBLEM 3 - Hard-Margin SVM with SciPy & Scikit-learn
# ============================================================

# ------------------------------------------------------------
#           SciPy Version
# ------------------------------------------------------------
print("===== Output from SciPy (Problem 3) =====\n")

# 1) Load and parse SAheart.data
data = np.genfromtxt("SAheart.data", delimiter=",", skip_header=1, dtype=None,
encoding=None)

with open("SAheart.data", "r") as f:
    headers = f.readline().strip().split(",")

feature_names = ["ldl", "adiposity", "typea", "obesity", "alcohol", "age"]
target_name = "chd"
feature_indices = [headers.index(fn) for fn in feature_names]
target_index = headers.index(target_name)
```

```
# 2) Only use first 20 samples
X = np.array([[float(row[i]) for i in feature_indices] for row in data[:20]])
y = np.array([1 if int(row[target_index]) == 1 else -1 for row in data[:20]])

# 3) SVM Dual
K = X @ X.T
P = np.outer(y, y) * K
n = X.shape[0]

def objective(alpha):
    return 0.5 * alpha @ P @ alpha - np.sum(alpha)

def svm_dual_constraint(alpha):
    return np.dot(alpha, y)

bounds = [(0, None)] * n
constraints = {"type": "eq", "fun": svm_dual_constraint}
alpha0 = np.zeros(n)

res = minimize(objective, alpha0, bounds=bounds, constraints=constraints)
alphas = res.x

# 4) Compute slope, intercept, etc.
w = ((alphas * y)[:, None] * X).sum(axis=0)
support_indices = np.where(alphas > 1e-5)[0]
```

```
b = np.mean([y[i] - np.dot(w, X[i]) for i in support_indices])

distances = (X @ w + b) / np.linalg.norm(w)
predictions = np.sign(X @ w + b)
accuracy = np.mean(predictions == y)
margin = 1.0 / np.linalg.norm(w)

x_new = np.ones(X.shape[1])  # All features=1
pred_label = np.sign(w @ x_new + b)

# 5) Print SciPy results
print("Dual variables (alphas):\n", alphas, "\n")
print("Slope vector (w):\n", w, "\n")
print("Intercept (b):\n", b, "\n")
print("Support vector indices:\n", support_indices, "\n")
print("Distances from hyperplane:\n", distances, "\n")
print("Accuracy:", accuracy, "\n")
print("Margin:", margin, "\n")
print("Prediction for x=all ones:", "CHD" if pred_label == 1 else "No CHD")



# ---------------------------------------------------------
#          Scikit-learn Version
# ---------------------------------------------------------
print("\n===== Output from Scikit-learn (Problem 3) =====\n")
```

```
clf = SVC(kernel="linear", C=1e10)  # approximate hard margin
clf.fit(X, y)

# (a) Reconstruct the dual variables alpha
#    scikit-learn only exposes alpha_i * y_i for SVs in clf.dual_coef_[0].
#    We place them into a length-n array, 0 for non-SVs.
alphas_skl = np.zeros(n, dtype=float)
alpha_y = clf.dual_coef_[0]       # shape: (1, n_SV)
support_indices_skl = clf.support_

for i, sv_ix in enumerate(support_indices_skl):
    alphas_skl[sv_ix] = alpha_y[i] / y[sv_ix]

w_skl = clf.coef_.ravel()
b_skl = clf.intercept_[0]
distances_skl = (X @ w_skl + b_skl) / np.linalg.norm(w_skl)
accuracy_skl = clf.score(X, y)
margin_skl = 1.0 / np.linalg.norm(w_skl)
pred_label_skl = np.sign(w_skl @ x_new + b_skl)

# Print scikit-learn results
print("Dual variables (alphas) reconstructed:\n", alphas_skl, "\n")
print("Slope vector (w):\n", w_skl, "\n")
print("Intercept (b):\n", b_skl, "\n")
print("Support vector indices:", support_indices_skl, "\n")
print("Distances from hyperplane:\n", distances_skl, "\n")
```

```
print("Accuracy:", accuracy_skl, "\n")
print("Margin:", margin_skl, "\n")
print("Prediction for x=all ones:", "CHD" if pred_label_skl == 1 else "No CHD")
```

a. **Dual variables:**

Dual variables (alphas):

[1.14414968e-03 1.79990124e-13 3.67738988e-13 8.85818122e-01
3.11377307e-14 1.27951034e+00 3.34267783e-13 1.80592981e-13
0.00000000e+00 8.85976642e-02 0.00000000e+00 1.12261754e+00
0.00000000e+00 0.00000000e+00 7.32206911e-01 2.15305828e-13
8.64602208e-02 1.43375356e-13 0.00000000e+00 6.78261849e-13]

b. **Slope parameter vector**

Slope vector (w):

[-0.09768246 0.76534166 0.2597387 -1.83248668 0.02896907 0.41584209]

c. **Intercept parameter**

Intercept (b):

-6.918286936334213

d. **Indices of support vectors**

Support vector indices:

[0 3 5 9 11 14 16]

e. **Distance of each data point from decision hyperplane and accuracy of classifier**

Distances from hyperplane:

[0.49505731 1.06775734 -1.56545379 0.49245178 1.9512953 -0.48659341

-0.86984256 0.91593522 -6.41665269 0.48610055 6.65894333 0.48269468

-8.07746514 -8.90511454 -0.49378332 -1.63742713 -0.48765922 1.00522143

 3.8602865   0.52646971]

Accuracy: 1.0

f. **Margin**

Margin: 0.4882683751314705

g. **Predict the label for all features being 1**

Prediction for x=all ones: No CHD

Problem 3 Part 2 Scikit-learn

a. **Dual variables**

Dual variables (alphas)

[1.15547129e-03 0.00000000e+00 0.00000000e+00 8.85864774e-01

0.00000000e+00 1.27962579e+00 0.00000000e+00 0.00000000e+00

0.00000000e+00 8.86169638e-02 0.00000000e+00 1.12272043e+00

0.00000000e+00 0.00000000e+00 7.32249135e-01 0.00000000e+00

8.64827114e-02 0.00000000e+00 0.00000000e+00 0.00000000e+00]

b. Slope vector w

Slope vector (w):

[-0.0979382 0.76487726 0.25990488 -1.83286612 0.02876446 0.41574002]

c. Intercept parameter

Intercept (b):

-6.893210350669595

d. Indices of the support vectors

e. Support vector indices: [5 14 16 0 3 9 11]

f. Distance from decision hyperplane and accuracy of classifier

Distances from hyperplane:

[0.48829007 1.06864007 -1.56468222 0.48812729 1.94851866 -0.48765153

-0.8628437   0.92140365 -6.41105977 0.48793703 6.65521481 0.48826814

-8.06709878 -8.8974895 -0.48837059 -1.63916091 -0.48798461 1.00456212

3.85465118 0.52487293]

Accuracy: 1.0

g. Margin

Margin: 0.48822648247479977

h. Prediction for all features = 1

Prediction for x=all ones: No CHD