```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import Lasso as SKLasso, Ridge as SKRidge, ElasticNet as
```

Question 1

```
a1 = np.array([4, 7, 9, 11, 15])
b1 = np.array([18, 22, 25, 30, 35, 40])
```

```
np.var(a1, ddof=1)
```

```
17.2
```

```
np.var(b1, ddof=1)
```

```
68.26666666666668
```

```
a2 = np.array([12, 14, 16, 18, 20])
b2 = np.array([28, 32, 36, 40, 44, 48])
```

```
np.var(a2, ddof=0)
```

```
8.0
```

```
np.var(b2, ddof=0)
```

```
46.666666666666664
```

Question 2

```
w0 = np.array([-1, 0, 1, 1]).reshape(-1, 1)
X = np.array([[1, -2, 0, 1],
              [-2, -1, 1, 2],
              [1, 2, -1, 1]])
Y = np.array([2, 3, -1]).reshape(-1, 1)
```

```
def d_f(w):
    return 2*X.T @ (X@w - Y) + 2*2*w
```

```
w1 = w0 - 0.5 * d_f(w0)
w1
```

```
array([[ 7.],
       [-2.],
       [-3.],
       [-3.]])
```

```
w2 = w1 - 0.5 * d_f(w1)
w2
```

```
array([[-65.],
       [-18.],
       [ 31.],
       [ 41.]])
```

Question 3

```
X = np.genfromtxt('stock_prediction_data.csv', delimiter=',')
y = np.genfromtxt('stock_price.csv', delimiter=',').reshape(-1, 1)
print(X.shape)
print(y.shape)
```

```
(300, 10)
(300, 1)
```

```
X_train, X_rest, y_train, y_rest = train_test_split(X, y, test_size=0.2, random_s
X_test, X_val, y_test, y_val = train_test_split(X_rest, y_rest, test_size=0.5, ra
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

X_train = PolynomialFeatures(degree=2, include_bias=True).fit_transform(X_train)
X_val = PolynomialFeatures(degree=2, include_bias=True).fit_transform(X_val)
X_test = PolynomialFeatures(degree=2, include_bias=True).fit_transform(X_test)
```

```
def mse(y_pred, y):
    return np.mean((y_pred - y)**2)
```

Lasso Constraint

My Gradient Descent

```
def lasso_d_f(X, w, y, λ):
    return 2/(X.shape[0]) * X.T @ (X @ w - y) + 2*λ*np.sign(w)


λ_list = [0, 0.25, 0.5, 0.75, 1, 1.5, 2, 3]
best_w = None
best_λ = None
best_MSE = None

for λ in λ_list:
    w = np.ones(X_train.shape[1]).reshape(-1, 1)

    for i in range(10000):
        w = w - 0.01 * lasso_d_f(X_train, w, y_train, λ)

    print("λ=" + str(λ) + " Validation MSE:", mse(X_val @ w, y_val))

    if best_MSE == None:
        best_w = w
        best_λ = λ
        best_MSE = mse(X_val @ w, y_val)
    elif mse(X_val @ w, y_val) < best_MSE:
        best_w = w
        best_λ = λ
        best_MSE = mse(X_val @ w, y_val)

print()
print("Best lambda value:", best_λ)
print("λ="+str(best_λ)+" Test MSE:", best_MSE)
```

```
λ=0 Validation MSE: 0.09294580769390001
λ=0.25 Validation MSE: 0.6481430052279473
λ=0.5 Validation MSE: 2.3135561251651287
λ=0.75 Validation MSE: 5.014351200194995
λ=1 Validation MSE: 7.729218232939105
λ=1.5 Validation MSE: 12.82838021897792
λ=2 Validation MSE: 20.3913536839422
λ=3 Validation MSE: 36.902216912470756

Best lambda value: 0
λ=0 Test MSE: 0.09294580769390001
```

Sklearn Regression

```
λ_list = [0, 0.25, 0.5, 0.75, 1, 1.5, 2, 3]
```

```
best_w = None
best_λ = None
best_MSE = None

for λ in λ_list:
    sk_poly_lasso = SKLasso(alpha=λ)
    sk_poly_lasso.fit(X_train,y_train.flatten()) # y is 2D, but scikit-learn expe
    pred_val = sk_poly_lasso.predict(X_val).reshape(-1,1)

    print("λ=" + str(λ) + " Validation MSE:", mse(y_val, pred_val))

    if best_MSE == None:
        best_w = w
        best_λ = λ
        best_MSE = mse(y_val, pred_val)
    elif mse(y_val, pred_val) < best_MSE:
        best_w = w
        best_λ = λ
        best_MSE = mse(y_val, pred_val)

print()
print("Best lambda value:", best_λ)
print("λ="+str(best_λ)+" Test MSE:", best_MSE)
```

```
λ=0 Validation MSE: 0.09294325090399339
λ=0.25 Validation MSE: 0.5848921507490163
λ=0.5 Validation MSE: 2.1051106412283676
λ=0.75 Validation MSE: 4.617968284978606
λ=1 Validation MSE: 7.629002980184824
λ=1.5 Validation MSE: 13.05233856483137
λ=2 Validation MSE: 20.535723261563653
λ=3 Validation MSE: 38.36305551717119

Best lambda value: 0
λ=0 Test MSE: 0.09294325090399339
/Users/jonanakai/anaconda3/envs/ds/lib/python3.12/site-packages/sklearn/base.p
  return fit_method(estimator, *args, **kwargs)
/Users/jonanakai/anaconda3/envs/ds/lib/python3.12/site-packages/sklearn/linea
  model = cd_fast.enet_coordinate_descent(
/Users/jonanakai/anaconda3/envs/ds/lib/python3.12/site-packages/sklearn/linea
  model = cd_fast.enet_coordinate_descent(
```

The MSE for my Lasso Constraint Gradient Descent code, approximately 0.093, was almost the same as the MSE for the Sklearn Lasso Constraint, approximately 0.093.

Ridge Constraint

My Gradient Descent

```
def ridge_d_f(X, w, y, λ):
    return 2*X.T @ (X@w - y) + 2*λ*w


λ_list = [0, 0.25, 0.5, 0.75, 1, 1.5, 2, 3]
best_w = None
best_λ = None
best_MSE = None

for λ in λ_list:
    w = np.ones(X_train.shape[1]).reshape(-1, 1)

    for i in range(10000):
        w = w - 0.0001 * ridge_d_f(X_train, w, y_train, λ)

    print("λ=" + str(λ) + " Validation MSE:", mse(X_val @ w, y_val))

    if best_MSE == None:
        best_w = w
        best_λ = λ
        best_MSE = mse(X_val @ w, y_val)
    elif mse(X_val @ w, y_val) < best_MSE:
        best_w = w
        best_λ = λ
        best_MSE = mse(X_val @ w, y_val)

print()
print("Best lambda value:", best_λ)
print("λ="+str(best_λ)+" Test MSE:", best_MSE)
```

```
λ=0 Validation MSE: 0.0929432509065546
λ=0.25 Validation MSE: 0.09632926686199482
λ=0.5 Validation MSE: 0.09996984059458619
λ=0.75 Validation MSE: 0.10385804104081169
λ=1 Validation MSE: 0.10798747846137201
λ=1.5 Validation MSE: 0.11694681755202548
λ=2 Validation MSE: 0.1268052376398113
λ=3 Validation MSE: 0.14907389969957377

Best lambda value: 0
λ=0 Test MSE: 0.0929432509065546
```

Sklearn Regression

```
λ_list = [0, 0.25, 0.5, 0.75, 1, 1.5, 2, 3]
best_w = None
best_λ = None
best_MSE = None
```

```
    _

for λ in λ_list:
    sk_poly_ridge = SKRidge(alpha=λ)
    sk_poly_ridge.fit(X_train,y_train.flatten()) # y is 2D, but scikit-learn expe
    pred_val = sk_poly_ridge.predict(X_val).reshape(-1,1)

    print("λ=" + str(λ) + " Validation MSE:", mse(y_val, pred_val))

    if best_MSE == None:
        best_w = w
        best_λ = λ
        best_MSE = mse(y_val, pred_val)
    elif mse(y_val, pred_val) < best_MSE:
        best_w = w
        best_λ = λ
        best_MSE = mse(y_val, pred_val)

print()
print("Best lambda value:", best_λ)
print("λ="+str(best_λ)+" Test MSE:", best_MSE)
```

```
λ=0 Validation MSE: 0.09149828125000004
λ=0.25 Validation MSE: 0.09531032106495477
λ=0.5 Validation MSE: 0.09792788380075418
λ=0.75 Validation MSE: 0.10079229994113074
λ=1 Validation MSE: 0.10389999417330338
λ=1.5 Validation MSE: 0.11083122633218859
λ=2 Validation MSE: 0.11869420111033205
λ=3 Validation MSE: 0.13711048894364736

Best lambda value: 0
λ=0 Test MSE: 0.09149828125000004
```

The MSE for my Ridge Constraint Gradient Descent code, approximately 0.093, was almost the same as the MSE for the Sklearn Ridge Constraint, approximately 0.091.

Elastic Net

My Gradient Descent

```
def elastic_d_f(X, w, y, λ):
    return 2*X.T @ (X@w - y) + 2*λ*np.sign(w) + 2*λ*w
```

```
λ_list = [0, 0.25, 0.5, 0.75, 1, 1.5, 2, 3]
best_w = None
best_λ = None
```

```
best_λ = None
best_MSE = None

for λ in λ_list:
    w = np.ones(X_train.shape[1]).reshape(-1, 1)

    for i in range(10000):
        w = w - 0.0001 * elastic_d_f(X_train, w, y_train, λ)

    print("λ=" + str(λ) + " Validation MSE:", mse(X_val @ w, y_val))

    if best_MSE == None:
        best_w = w
        best_λ = λ
        best_MSE = mse(X_val @ w, y_val)
    elif mse(X_val @ w, y_val) < best_MSE:
        best_w = w
        best_λ = λ
        best_MSE = mse(X_val @ w, y_val)

print()
print("Best lambda value:", best_λ)
print("λ="+str(best_λ)+" Test MSE:", best_MSE)
```

```
λ=0 Validation MSE: 0.0929432509065546
λ=0.25 Validation MSE: 0.09266365424469683
λ=0.5 Validation MSE: 0.0924768241225078
λ=0.75 Validation MSE: 0.09240428564745783
λ=1 Validation MSE: 0.09347114200230527
λ=1.5 Validation MSE: 0.09583804791164616
λ=2 Validation MSE: 0.09825181950067377
λ=3 Validation MSE: 0.1067080433573084

Best lambda value: 0.75
λ=0.75 Test MSE: 0.09240428564745783
```

Sklearn Regression

```
λ_list = [0, 0.25, 0.5, 0.75, 1, 1.5, 2, 3]
best_w = None
best_λ = None
best_MSE = None

for λ in λ_list:
    sk_poly_elastic = SKElastic(alpha=λ)
    sk_poly_elastic.fit(X_train,y_train.flatten()) # y is 2D, but scikit-learn ex
    pred_val = sk_poly_elastic.predict(X_val).reshape(-1,1)

    print("λ=" + str(λ) + " Validation MSE:", mse(y_val, pred_val))
```

```
    if best_MSE == None:
        best_w = w
        best_λ = λ
        best_MSE = mse(y_val, pred_val)
    elif mse(y_val, pred_val) < best_MSE:
        best_w = w
        best_λ = λ
        best_MSE = mse(y_val, pred_val)

print()
print("Best lambda value:", best_λ)
print("λ="+str(best_λ)+" Test MSE:", best_MSE)
```

```
λ=0 Validation MSE: 0.09294325090399339
λ=0.25 Validation MSE: 1.3093410881089398
λ=0.5 Validation MSE: 3.995272505431218
λ=0.75 Validation MSE: 7.311109090036272
λ=1 Validation MSE: 10.827746499666107
λ=1.5 Validation MSE: 17.722082297295813
λ=2 Validation MSE: 23.9603694212866
λ=3 Validation MSE: 32.984971725551866

Best lambda value: 0
λ=0 Test MSE: 0.09294325090399339
/Users/jonanakai/anaconda3/envs/ds/lib/python3.12/site-packages/sklearn/base.
  return fit_method(estimator, *args, **kwargs)
/Users/jonanakai/anaconda3/envs/ds/lib/python3.12/site-packages/sklearn/linea
  model = cd_fast.enet_coordinate_descent(
/Users/jonanakai/anaconda3/envs/ds/lib/python3.12/site-packages/sklearn/linea
  model = cd_fast.enet_coordinate_descent(
```

The MSE for my Elastic Net Gradient Descent code, approximately 0.093, was almost the same as the MSE for the Sklearn Elastic Net, approximately 0.093.

The best lmabda value for each constraint was 0, making the contraint function irrelevant. If we compare the constraint functions for nonzero lambda values, for example 1, we get Lasso: 7.63, Ridge: 0.10, and Elastic: 10.83. So Elasitic Net appears to perform worse than using only one constraint.