Lab 4 solution key

1.

```
import numpy as np

# Sample Data
x = np.array([1, 2, 3, 4, 5], dtype=float)
y = np.array([2, 3, 5, 6, 8], dtype=float)

# Calculate slope (m) and intercept (c)
x_mean = np.mean(x)
y_mean = np.mean(y)

m = np.sum((x - x_mean) * (y - y_mean)) / np.sum((x - x_mean) ** 2)
c = y_mean - m * x_mean

print(f"Least Squares Slope (m): {m:.4f}")
print(f"Least Squares Intercept (c): {c:.4f}")
```

2.

```
import numpy as np
from sklearn.datasets import load_diabetes
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# --- (a) Synthetic Data Generation & Training ---
np.random.seed(42)
X_syn = np.random.rand(150, 1) * 10
noise = np.random.randn(150, 1) * 2
y_syn = 2.5 * X_syn + 3 + noise

X_train, X_test, y_train, y_test = train_test_split(
    X_syn, y_syn, test_size=0.2, random_state=42
)

model_syn = LinearRegression()
model_syn.fit(X_train, y_train)

print(
    f"(a) Coefficient: {model_syn.coef_[0][0]:.4f}, Intercept: {model_syn.intercept_[0]:.4f}"
)

# --- (b) Evaluation Metrics ---
```

```
def evaluate(y_true, y_pred, set_name):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    print(
        f"[{set_name}] MSE: {mse:.4f} | RMSE: {rmse:.4f} | MAE: {mae:.4f} | R2: {r2:.4f}"
    )


print("\n(b) Synthetic Data Metrics:")
evaluate(y_train, model_syn.predict(X_train), "Train")
evaluate(y_test, model_syn.predict(X_test), "Test")

# --- (c) Diabetes Dataset (BMI Feature) ---
diabetes = load_diabetes()
X_bmi = diabetes.data[:, [2]]  # Column index 2: BMI
y_diab = diabetes.target

X_tr_d, X_te_d, y_tr_d, y_te_d = train_test_split(
    X_bmi, y_diab, test_size=0.2, random_state=42
)
model_diab = LinearRegression()
model_diab.fit(X_tr_d, y_tr_d)

print("\n(c) Diabetes (BMI Feature) Metrics:")
evaluate(y_te_d, model_diab.predict(X_te_d), "Diabetes Test")
print(
    f"Interpretation: BMI Coefficient is {model_diab.coef_[0]:.2f}, indicating a positive relation with
disease progression."
)
```

3.

```
import numpy as np
from sklearn.datasets import fetch_california_housing
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# --- 3(a) Normal Equation Implementation ---
X = np.array([[1, 2], [2, 1], [3, 4], [4, 3]])
y = np.array([6, 5, 11, 10])
```

```
# Add bias column of ones
X_b = np.c_[np.ones((X.shape[0], 1)), X]
beta_normal = np.linalg.inv(X_b.T @ X_b) @ X_b.T @ y
print("3(a) Normal Equation Coefficients (Intercept, B1, B2):\n", beta_normal)


# --- 3(a-2) Gradient Descent Class ---
class LinearRegressionGD:

    def __init__(self, lr=0.01, iterations=1000):
        self.lr = lr
        self.iterations = iterations

    def fit(self, X, y):
        m, n = X.shape
        self.weights = np.zeros(n)
        self.bias = 0.0

        for _ in range(self.iterations):
            y_pred = X @ self.weights + self.bias
            dw = (1 / m) * (X.T @ (y_pred - y))
            db = (1 / m) * np.sum(y_pred - y)

            self.weights -= self.lr * dw
            self.bias -= self.lr * db

    def predict(self, X):
        return X @ self.weights + self.bias


# Quick check of GD Class
gd_clf = LinearRegressionGD(lr=0.01, iterations=2000)
gd_clf.fit(X, y)
print(f"\n3(a-2) GD Class Fitted -> Bias: {gd_clf.bias:.4f}, Weights: {gd_clf.weights}")


# --- 3(b) California Housing Dataset with Scikit-Learn ---
housing = fetch_california_housing()
X_h, y_h = housing.data, housing.target

X_tr_h, X_te_h, y_tr_h, y_te_h = train_test_split(
    X_h, y_h, test_size=0.2, random_state=42
)
```

```
scaler = StandardScaler()
X_tr_h_scaled = scaler.fit_transform(X_tr_h)
X_te_h_scaled = scaler.transform(X_te_h)

sk_housing_model = LinearRegression()
sk_housing_model.fit(X_tr_h_scaled, y_tr_h)

y_pred_h = sk_housing_model.predict(X_te_h_scaled)
print("\n3(b) California Housing Dataset Evaluation:")
print(f"MSE: {mean_squared_error(y_te_h, y_pred_h):.4f}")
print(f"R² Score: {r2_score(y_te_h, y_pred_h):.4f}")
```