Part a)

# question 1 - part a

# read the dataset as a set of arrays.

import numpy as np

from typing import Tuple

def load\_dataset(src\_dir: str) -> Tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray, np.ndarray]:

    x\_train = np.loadtxt(src\_dir + 'x\_train.txt')

    y\_train = np.loadtxt(src\_dir + 'y\_train.txt')

    x\_val = np.loadtxt(src\_dir + 'x\_val.txt')

    y\_val = np.loadtxt(src\_dir + 'y\_val.txt')

    x\_test = np.loadtxt(src\_dir + 'x\_test.txt')

    return x\_train, y\_train, x\_val, y\_val, x\_test

Part b) i.

# n - hyper parameter

# each w is a weight

# question 1 part b sub part a

# make input features for the above linear regression model.

def get\_features(x: np.ndarray, n: int) -> np.ndarray:

    features = []

    for i in range(1,n+1):

        features.append(np.power(x,i))

    features\_nparray = np.array(features)

    features\_output = np.transpose(features\_nparray)

    return features\_output

Part b) ii.

# question 1 part b sub part 2

# fit and evaluate function

from sklearn.linear\_model import LinearRegression

# from sklearn.metrics import mean\_squared\_error # this can be achieved using np.mean function

def fit\_and\_evaluate(x\_train:np.ndarray, y\_train:np.ndarray, x\_val:np.ndarray, y\_val:np.ndarray, n:int) -> Tuple[float, float]:

    regressor = LinearRegression()

    x\_train\_features = get\_features(x\_train, n)

    x\_val\_features = get\_features(x\_val, n)

    regressor.fit(x\_train\_features,y\_train)

    # W = regressor.coef\_

    y\_predict\_train = regressor.predict(x\_train\_features)

    y\_predict\_val = regressor.predict(x\_val\_features)

    train\_mse = np.mean((y\_predict\_train - y\_train)\*\*2)

    val\_mse = np.mean((y\_predict\_val - y\_val)\*\*2)

    return train\_mse, val\_mse

Chart, line chart

Description automatically generatedPart c)

Part d)

Since the validation MSE is least at n=4 we chose it for prediction on test set.