# Question 1 code

```

# # 1 Linear Regressor


# ## (b)

import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


class regressor:
    def __init__(self, feature_dim, a, b):
        self.weight = np.random.uniform(low=-0.1, high=0.1,
size=feature_dim)
        self.lr = 0
        self.a = a
        self.b = b

    def train(self, X, y, epochs=100):
        self.errors = []
        error0 = self.cal_rse(X, y)
        self.errors.append(error0)
        ## Gradient descent
        for epoch in range(epochs):
            for i in range(X.shape[0]):
                self.schedule_lr(i + epoch * X.shape[0])
                gradient = 2 / X.shape[0] * (np.dot(self.weight, X[i]) -
y[i]) * X[i]
                self.weight -= self.lr * gradient
            error = self.cal_rse(X, y)
            if error < 0.001 * error0:
                break
            self.errors.append(error)
        return

    def cal_rse(self, X, y):
        error = np.sqrt(np.mean((X @ self.weight - y)**2))
        return error

    def predict(self, X):
        y_pred = X @ self.weight
        return y_pred

    def schedule_lr(self, i):
        self.lr = self.a / (i + self.b)


df_train = pd.read_csv("./h5w7_pr1_power_train.csv")
```

```
df_test = pd.read_csv("./h5w7_pr1_power_test.csv")
X_train = df_train.loc[:, ['AT', 'V', 'AP', 'RH']].values
X_train = np.concatenate((X_train, np.ones((X_train.shape[0], 1))), axis=1)
y_train = df_train.loc[:, ['PE']].values.ravel()
X_test = df_test.loc[:, ['AT', 'V', 'AP', 'RH']].values
X_test = np.concatenate((X_test, np.ones((X_test.shape[0], 1))), axis=1)
y_test = df_test.loc[:, ['PE']].values.ravel()
X_train.shape, y_train.shape, X_test.shape, y_test.shape


A = [0.01, 0.1, 1, 10, 100]
B = [1, 10, 100, 1000]
weights_dict = {}
errors_dict = {}
for a in A:
    for b in B:
        reg = regressor(feature_dim=5, a=a, b=b)
        reg.train(X_train, y_train)
        weights_dict['{},{}'.format(a, b)] = reg.weight
        errors_dict['{},{}'.format(a, b)] = reg.errors



for a in A:
    _ = plt.figure(figsize=(8,6))
    for b in B:
        errors = errors_dict['{},{}'.format(a, b)]
        _ = plt.plot(list(range(len(errors))), errors, label='b='+str(b))
    _ = plt.xlabel('Epoch')
    _ = plt.ylabel('Root Mean Square Error')
    _ = plt.title('Root Mean Square Error via different b (a=
{})'.format(a))
    _ = plt.legend()
    plt.savefig("./figs/p1_a_{}.png".format(a), dpi=300)
    plt.show()


# ## (d)


a, b = 100, 1
y_pred = X_test @ weights_dict['{},{}'.format(a, b)]
rse_best = np.sqrt(np.mean((y_pred - y_test)**2))
print("The best rse is {:.3f}".format(rse_best))


# ## (e)


y_pred_trival = np.mean(y_train)
rse_trival = np.sqrt(np.mean((y_pred_trival - y_test)**2))
print("The trival rse is {:.3f}".format(rse_trival))
```