

```

import numpy as np
import tensorflow as tf
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.utils import resample

# Load Fashion MNIST dataset
fashion_mnist = tf.keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

# Normalize the pixel values to be between 0 and 1
X_train, X_test = X_train / 255.0, X_test / 255.0

# Flatten the images to make them compatible with logistic regression
X_train = X_train.reshape(X_train.shape[0], -1)
X_test = X_test.reshape(X_test.shape[0], -1)

# Convert labels to binary (for simplicity, let's classify between two classes: 0 (T-shirt/top) and 1 (Trouser))
binary_indices_train = np.where((y_train == 0) | (y_train == 1))[0]
binary_indices_test = np.where((y_test == 0) | (y_test == 1))[0]

X_train_binary, y_train_binary = X_train[binary_indices_train], y_train[binary_indices_train]
X_test_binary, y_test_binary = X_test[binary_indices_test], y_test[binary_indices_test]

# Convert labels to 0 and 1
y_train_binary = np.where(y_train_binary == 0, 0, 1)
y_test_binary = np.where(y_test_binary == 0, 0, 1)

# Define sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Function to train the model and return predictions
def train_and_predict(X_train, y_train, X_test):
    m = np.zeros(X_train.shape[1])
    c = 0
    LR = 0.0001
    epochs = 50

    for epoch in range(1, epochs + 1):
        for i in range(len(X_train)):
            gr_wrt_m = X_train[i] * (y_train[i] - sigmoid(np.dot(m.T, X_train[i]) + c))
            gr_wrt_c = y_train[i] - sigmoid(np.dot(m.T, X_train[i]) + c)
            m = m + LR * gr_wrt_m
            c = c + LR * gr_wrt_c

    predictions = []
    for i in range(len(X_test)):
        z = np.dot(m, X_test[i]) + c
        y_pred = sigmoid(z)
        predictions.append(y_pred)
    return np.array(predictions)

# Train multiple models and collect their predictions
n_models = 10
all_predictions = []

for _ in range(n_models):
    X_resampled, y_resampled = resample(X_train_binary, y_train_binary)
    predictions = train_and_predict(X_resampled, y_resampled, X_test_binary)
    all_predictions.append(predictions)

all_predictions = np.array(all_predictions)


# Calculate the average prediction
average_prediction = np.mean(all_predictions, axis=0)

# Calculate bias
bias = mean_squared_error(y_test_binary, average_prediction)

# Calculate variance
variance = np.mean(np.var(all_predictions, axis=0))

# Output bias and variance
print("Bias:", bias)
print("Variance:", variance)

```

 Bias: 0.014884992836837167  
 Variance: 8.482621657344636e-05

