Lab 2: Using Scikit and Numpy for MSE Calculations (11/09/2024)

Matthew Loh

Table of contents

1 Define Training Data **1**
1.1 2. Check whether MSE = 0.21606 2
1.2 3. There are TWO ways to automatically calculate MSE: 3
1.2.1 a) Using scikit learn 3
1.2.2 b) MSE using Numpy module 3
1.3 Previous Lab Code 3
1.3.1 Trials 5

```
import numpy as np
import matplotlib.pyplot as plt
```

1 Define Training Data

Lab 3 Consider the given data points: (1,1), (2,1), (3,2), (4,2), (5,4). Regression line equation: Y = 0.7X – 0.1 ## 1. Check whether the following is correct:

X	Y	y hat
1	1	0.6
2	1	1.29
3	2	1.99
4	2	2.69
5	4	3.4

```
# Calculating from the equation
def calculate_y_hat(x):
    return 0.7 * x - 0.1


x = [1, 2, 3, 4, 5]
y = [1, 1, 2, 2, 4]
y_hat = [calculate_y_hat(x[i]) for i in range(len(x))]

print(y_hat)

# [0.6, 1.2999999999999998, 1.9999999999999996, 2.6999999999999997, 3.4]
print("The values are correct")
```

```
[0.6, 1.2999999999999998, 1.9999999999999996, 2.6999999999999997, 3.4]
The values are correct
```

1.1 2. Check whether MSE = 0.21606

```
# Calculating MSE
def calculate_mse(y, y_hat):
    return np.mean((np.array(y) - np.array(y_hat)) ** 2)


def calculate_mse_array_inputs(y, y_hat):
    return np.mean((y - y_hat) ** 2)


print(calculate_mse(y, y_hat))
# Lab values
# Given values
Y_true = [1, 1, 2, 2, 4]  # Y_true = Y (original values)
# calculated values
Y_pred = [0.6, 1.29, 1.99, 2.69, 3.4]  # Y_pred = Y'
# Calculation of Mean Squared Error (MSE)
print(calculate_mse(Y_true, Y_pred))
```

```
0.21999999999999992
0.21606
```

1.2 3. There are TWO ways to automatically calculate MSE:

1.2.1 a) Using scikit learn

```
from sklearn.metrics import mean_squared_error

# Given values
Y_true = [1, 1, 2, 2, 4]  # Y_true = Y (original values)
# calculated values
Y_pred = [0.6, 1.29, 1.99, 2.69, 3.4]  # Y_pred = Y'
# Calculation of Mean Squared Error (MSE)
mean_squared_error(Y_true, Y_pred)
```

```
0.21606
```

1.2.2 b) MSE using Numpy module

```
import numpy as np
# Given values
Y_true = [1, 1, 2, 2, 4]  # Y_true = Y (original values)
# calculated values
Y_pred = [0.6, 1.29, 1.99, 2.69, 3.4]  # Y_pred = Y'
# Mean squared Error
MSE = np.square(np.subtract(Y_true, Y_pred)).mean()
MSE
```

```
0.21606
```

4. Use both functions to calculate MSE for the different W_s and b_s in your previous lab code from last week

1.3 Previous Lab Code

```
# From previous lab code
# x_train is the input variable (size in 1000 square feet)
# y_train is the target (price in 1000s of dollars)
x_train = np.array([1.0, 2.0])
```

```
y_train = np.array([300.0, 500.0])
print(f"x_train: {x_train}")
print(f"y_train: {y_train}")


def compute_model_output(x: np.ndarray, w: float, b: float) -> np.ndarray:
    """
    Computes the prediction of a linear model
    Args:
      x (np.ndarray (m,)): Data, m examples
      w,b (scalar) : Model parameters
    Returns:
      y (ndarray (m,)) : target values
    """
    m = x.shape[0]
    f_wb = np.zeros(m)
    for i in range(m):
        f_wb[i] = w * x[i] + b
    return f_wb


from sklearn.metrics import mean_squared_error
import numpy as np


def scikit_learn_mse(y_true, y_pred):
    return mean_squared_error(y_true, y_pred)


def numpy_mse(y_true, y_pred):
    return np.square(np.subtract(y_true, y_pred)).mean()


def output(x_train, y_train, w, b):
    y_pred = compute_model_output(x_train, w, b)
    print(f"y_pred: {y_pred}")
    print(f"scikit learn mse: {scikit_learn_mse(y_train, y_pred)}")
    print(f"numpy mse: {numpy_mse(y_train, y_pred)}")


print(f"y_true: {y_train}")
```

```
x_train: [1. 2.]
y_train: [300. 500.]
y_true: [300. 500.]
```

1.3.1 Trials

```
w = 100
b = 100
output(x_train, y_train, w, b)
```

```
y_pred: [200. 300.]
scikit learn mse: 25000.0
numpy mse: 25000.0
```

```
w = 150
b = 100
output(x_train, y_train, w, b)
```

```
y_pred: [250. 400.]
scikit learn mse: 6250.0
numpy mse: 6250.0
```

```
w = 50
b = 100
output(x_train, y_train, w, b)
```

```
y_pred: [150. 200.]
scikit learn mse: 56250.0
numpy mse: 56250.0
```

```
w = 200
b = 100
output(x_train, y_train, w, b)
```

```
y_pred: [300. 500.]
scikit learn mse: 0.0
numpy mse: 0.0
```

```
w = 198
b = 102.5
output(x_train, y_train, w, b)
```

```
y_pred: [300.5 498.5]
scikit learn mse: 1.25
numpy mse: 1.25
```