

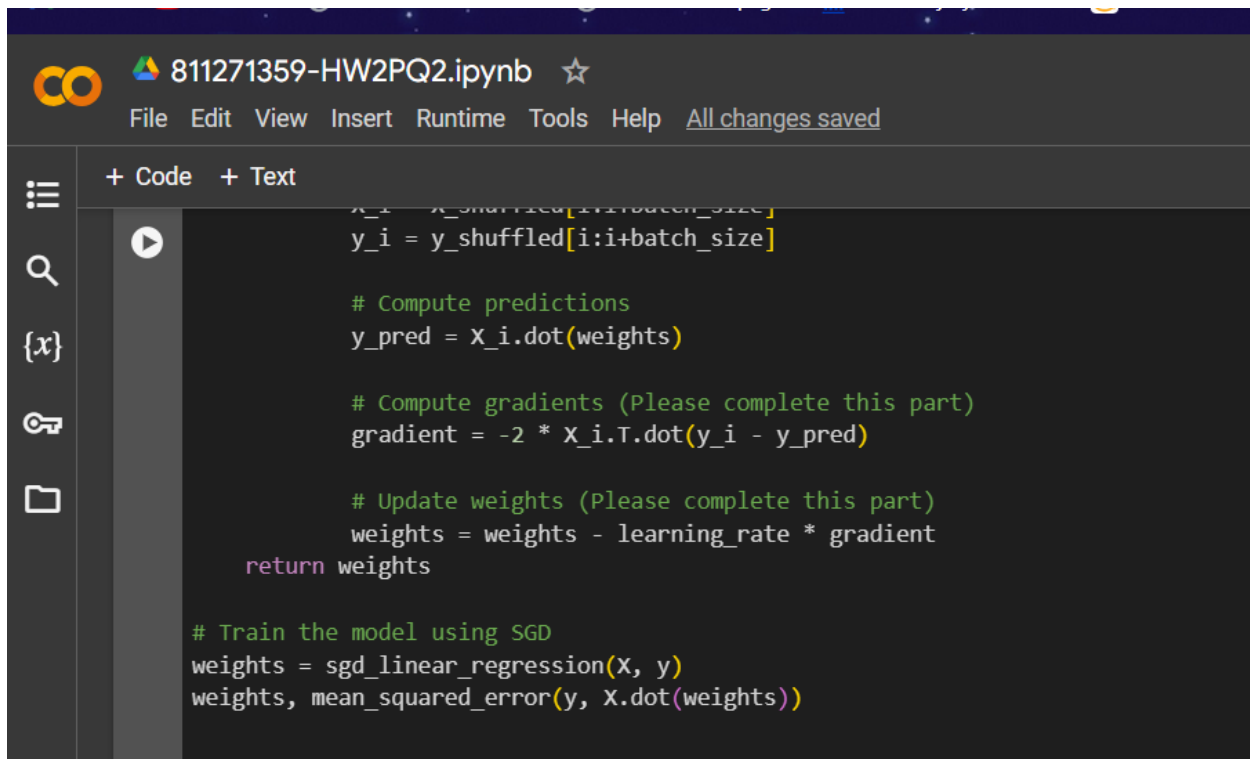
Name : Abhishek Milind Patwardhan

ID : 811271359

Data Science - II

Homework - 2

Changes in Lines of Code of Q1-2) Linear regression.py File



The screenshot shows a Jupyter Notebook titled "811271359-HW2PQ2.ipynb". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", "Help", and "All changes saved". On the left, there is a sidebar with icons for a list, search, variables, keys, and files. The main area displays the following Python code:

```
x_i = x_shuffled[i:i+batch_size]
y_i = y_shuffled[i:i+batch_size]

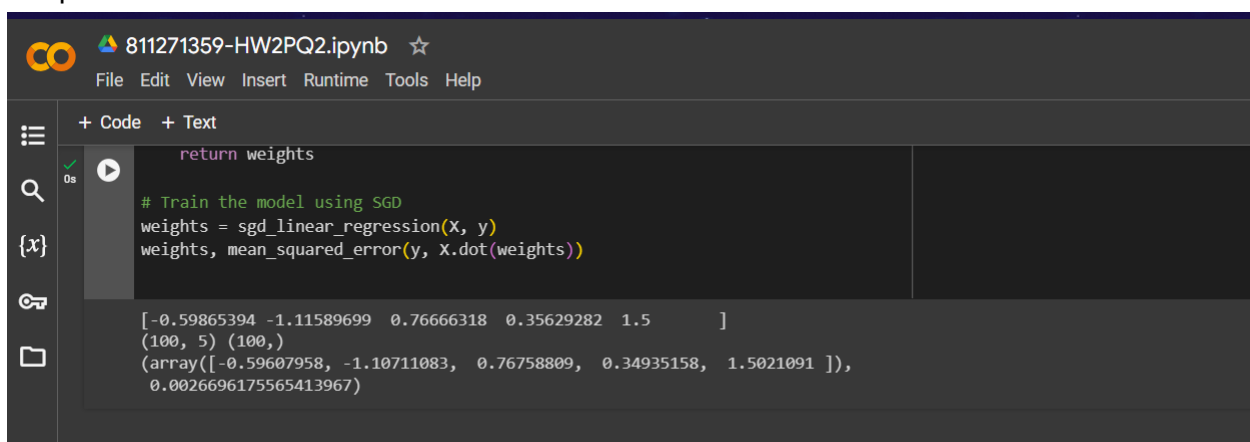
# Compute predictions
y_pred = X_i.dot(weights)

# Compute gradients (Please complete this part)
gradient = -2 * X_i.T.dot(y_i - y_pred)

# Update weights (Please complete this part)
weights = weights - learning_rate * gradient
return weights

# Train the model using SGD
weights = sgd_linear_regression(X, y)
weights, mean_squared_error(y, X.dot(weights))
```

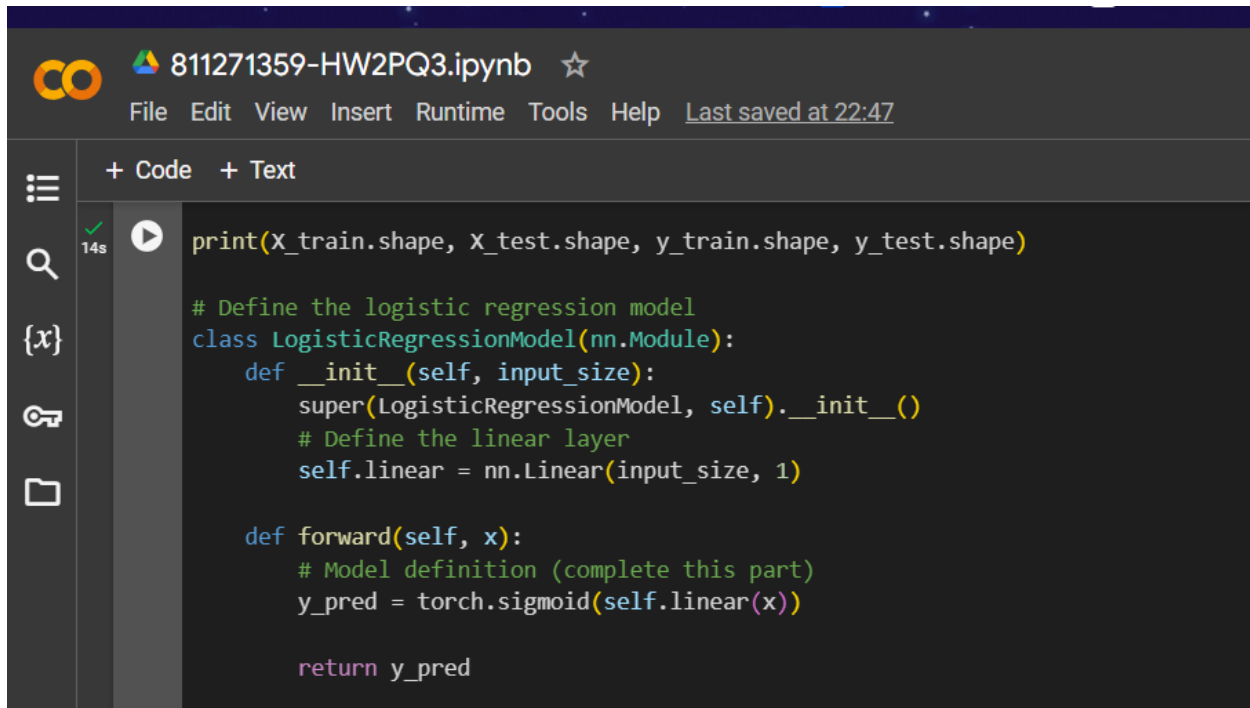
Output :



The screenshot shows the same Jupyter Notebook interface, but now it displays the output of the code. The code cell is executed, and the output is shown in a separate cell below it. The output is a tuple containing the weights and the mean squared error:

```
[ -0.59865394 -1.11589699  0.76666318  0.35629282  1.5          ]
(100, 5) (100,)
(array([ -0.59607958, -1.10711083,  0.76758809,  0.34935158,  1.5021091 ]),
 0.0026696175565413967)
```

Changes in Lines of Code of Q1-3) Logistic Regression.py File



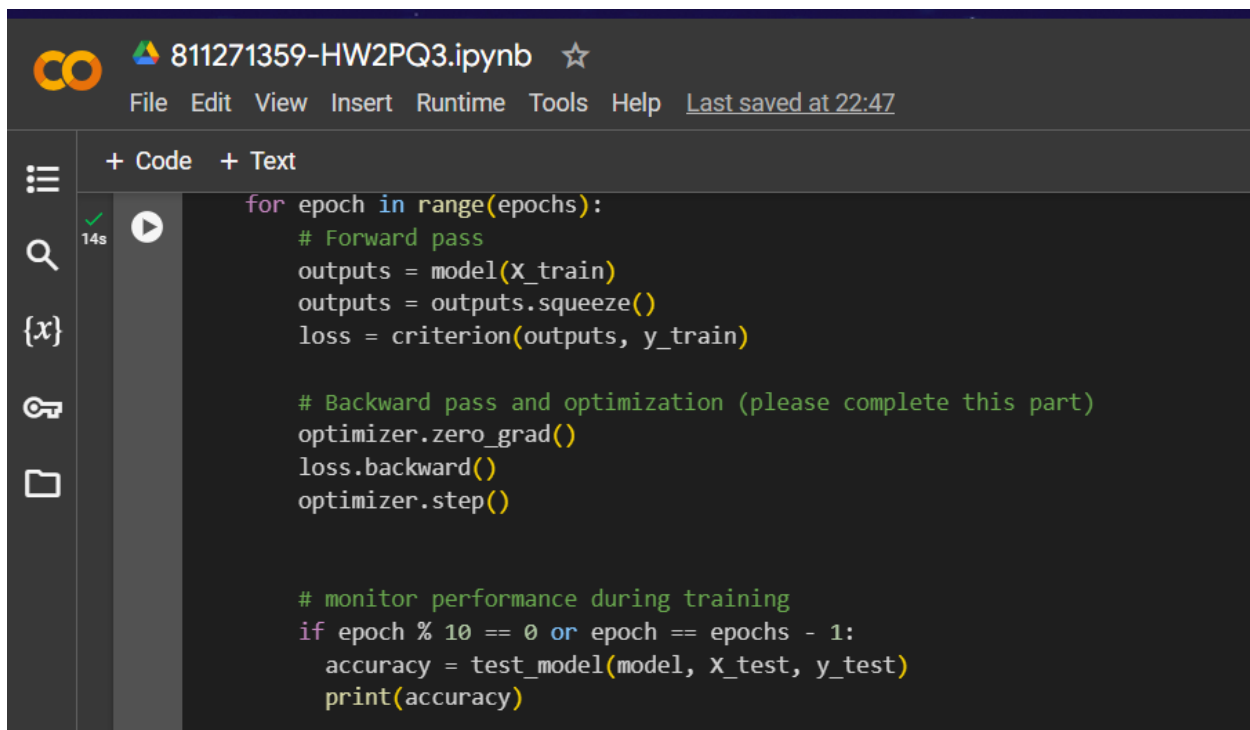
The screenshot shows a Jupyter Notebook titled "811271359-HW2PQ3.ipynb". The interface includes a top bar with a file explorer icon, the title, and a star icon. Below the title is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The main area is divided into a left sidebar with icons for file explorer, search, and code execution, and a central code editor. The code editor shows the following Python code:

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

# Define the logistic regression model
class LogisticRegressionModel(nn.Module):
    def __init__(self, input_size):
        super(LogisticRegressionModel, self).__init__()
        # Define the linear layer
        self.linear = nn.Linear(input_size, 1)

    def forward(self, x):
        # Model definition (complete this part)
        y_pred = torch.sigmoid(self.linear(x))

        return y_pred
```



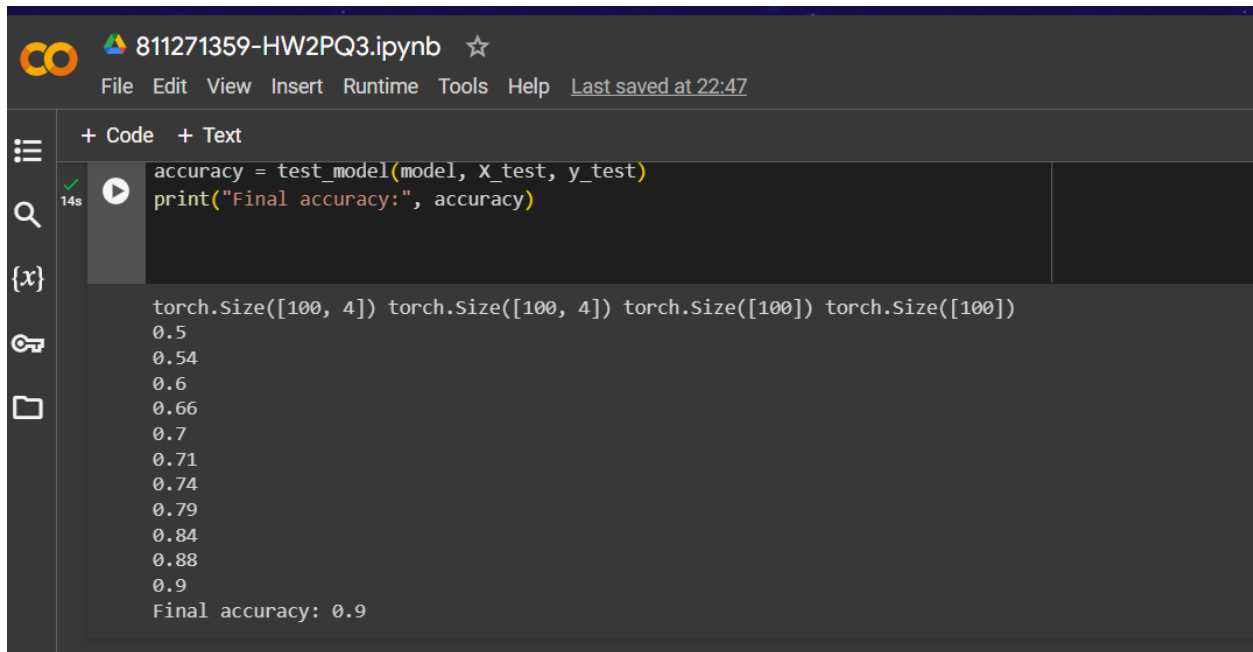
The screenshot shows the same Jupyter Notebook interface as the previous one, but with the following Python code in the code editor:

```
for epoch in range(epochs):
    # Forward pass
    outputs = model(X_train)
    outputs = outputs.squeeze()
    loss = criterion(outputs, y_train)

    # Backward pass and optimization (please complete this part)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # monitor performance during training
    if epoch % 10 == 0 or epoch == epochs - 1:
        accuracy = test_model(model, X_test, y_test)
        print(accuracy)
```

Output :



The image shows a Jupyter Notebook interface with a dark theme. At the top, the notebook is titled "811271359-HW2PQ3.ipynb" with a star icon. Below the title is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a link "Last saved at 22:47". On the left side, there is a sidebar with icons for a menu, search, variables, runtime, and files. The main area is divided into two sections: a code editor and an output area. The code editor contains two lines of Python code: `accuracy = test_model(model, X_test, y_test)` and `print("Final accuracy:", accuracy)`. The output area shows the results of the code execution, which are a series of accuracy values: 0.5, 0.54, 0.6, 0.66, 0.7, 0.71, 0.74, 0.79, 0.84, 0.88, 0.9, and finally "Final accuracy: 0.9".

```
accuracy = test_model(model, X_test, y_test)
print("Final accuracy:", accuracy)
```

0.5
0.54
0.6
0.66
0.7
0.71
0.74
0.79
0.84
0.88
0.9
Final accuracy: 0.9