HW6

Comparison of HW4 and Linear Regression

Linear Regression: MAE = 49.16
1-hidden layer Neural Network: MAE = 41.81

The MAE measures the average absolute difference between predicted and actual values. In this case, the lower MAE is better as it indicates that the model's predictions are, on average, closer to the actual values.

This suggests that the neural network model has better predictive performance in terms of capturing the underlying patterns in the data.

The neural network's advantage likely comes from its ability to model non-linear relationships between input features and the target variable. It can capture more complex patterns that Linear Regression, which assumes a linear relationship, might not capture as effectively.

However, it's important to note that the neural network is a more complex model, and its performance can be influenced by factors like architecture, hyperparameters, and the amount of data available for training.

In summary, based on the provided MAE values, the 1-hidden layer Neural Network appears to be the better-performing model in this particular regression task. It has a lower MAE, indicating that it provides predictions that are, on average, closer to the true values. However, it's essential to consider the context of your specific problem and other factors like model interpretability and computational cost when choosing between these two approaches.

.

HW6

Comparison of HW5 and Logistic Regression

Test Accuracy

Logistic Regression: 85.00%
1-hidden layer Neural Network (6-6-3 architecture): 75.00%

In logistic regression, we have to use stratification while splitting the data and appropriate hyperparameters while training the data.

Because we have 3 categories of high, medium and low the stratification will help to keep the same ratio of these categories in both training and testing datasets.

```
# Split the data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X_new, y_new, test_size=0.2, random_state=42,stratify = y_new)

# Create a Logistic Regression model and fit it to the training data
#logistic_regression_model = LogisticRegression(max_iter=1000)
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train, y_train)

# Evaluate the model on the test data
y_pred = logistic_regression_model.predict(X_test)
```

Parameter of multi_class = ‘multinomial’ and compatible solver types such as ‘lbgfs’, ‘sag’ and ‘saga’ will get us more accurate results than default parameters.

The max_iter parameter in logistic regression specifies the maximum number of iterations that the solver should run to find the optimal solution. It's used to control the convergence of the optimization algorithm. When we set max_iter to 1000, you are telling the logistic regression solver to perform a maximum of 1000 iterations to find the best model parameters. So I tried different values for max_iter and when the value is set to 425 the accuracy increased to 85%.

```
# Split the data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X_new, y_new, test_size=0.2, random_state=42,stratify = y_new)

# Create a Logistic Regression model and fit it to the training data
#logistic_regression_model = LogisticRegression(max_iter=1000)
logistic_regression_model = LogisticRegression(solver = 'lbfgs',multi_class='multinomial', max_iter=425)
logistic_regression_model.fit(X_train, y_train)

# Evaluate the model on the test data
y_pred = logistic_regression_model.predict(X_test)
```

[13] ✓ 0.0s Python

It's important to note that test accuracy is just one metric, and the choice between these models might depend on other factors such as model complexity, interpretability, and the nature of the dataset. Logistic regression is a linear model, and it seems to perform well for this task, which suggests that the relationship between input features and the target categories may be linear or relatively simple.

The neural network, with its non-linear activation functions and multiple layers, is more complex and might capture more intricate patterns, but it didn't perform as well in this case. You could explore different neural network architectures, hyperparameters, or additional data preprocessing techniques to potentially improve the neural network's performance.