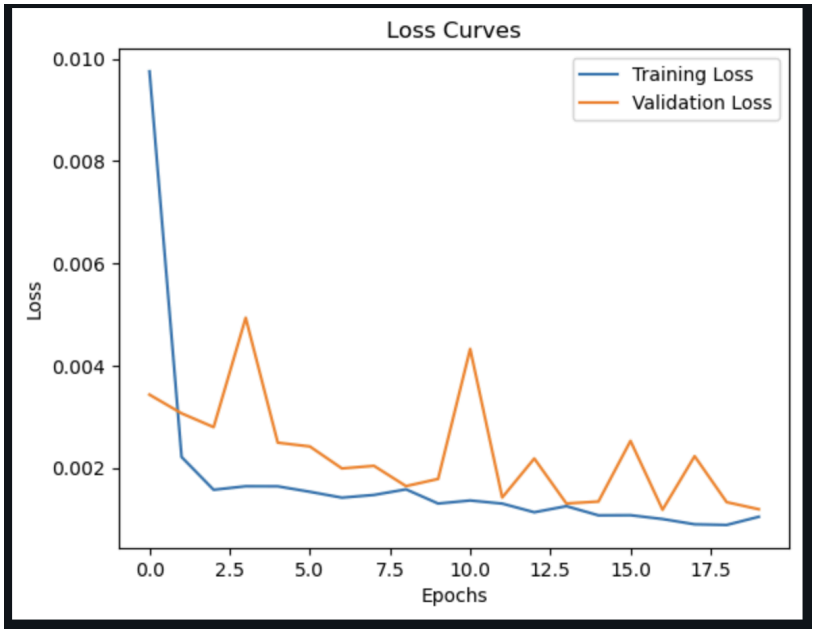
LSTM Model Development

1. Steps Taken to Build the LSTM Model:

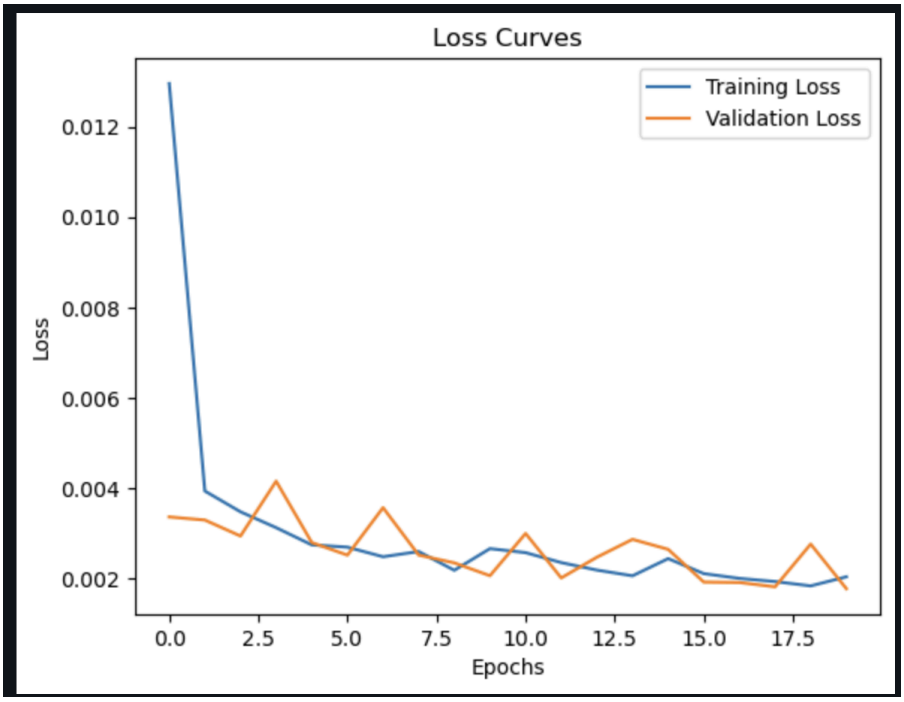
- Data Preparation:
  - Collected and processed historical TSLA stock data, focusing initially on the 'Adj Close' column for price prediction.
- Model Architecture:
  - Created an LSTM model with the following layers:
    - LSTM layer to capture time dependencies.
    - Dropout layer to prevent overfitting by randomly dropping some neurons during training.
    - Dense output layer to predict the adjusted closing prices.
- Loss Function & Optimization:
  - Used Mean Squared Error (MSE) as the loss function, with Adam optimizer for efficient gradient-based optimization.

2. Decision-Making and Challenges:

- Overfitting Problem:
  - After initial model training, I observed a significant amount of overfitting. The model performed well on training data but poorly on validation data.



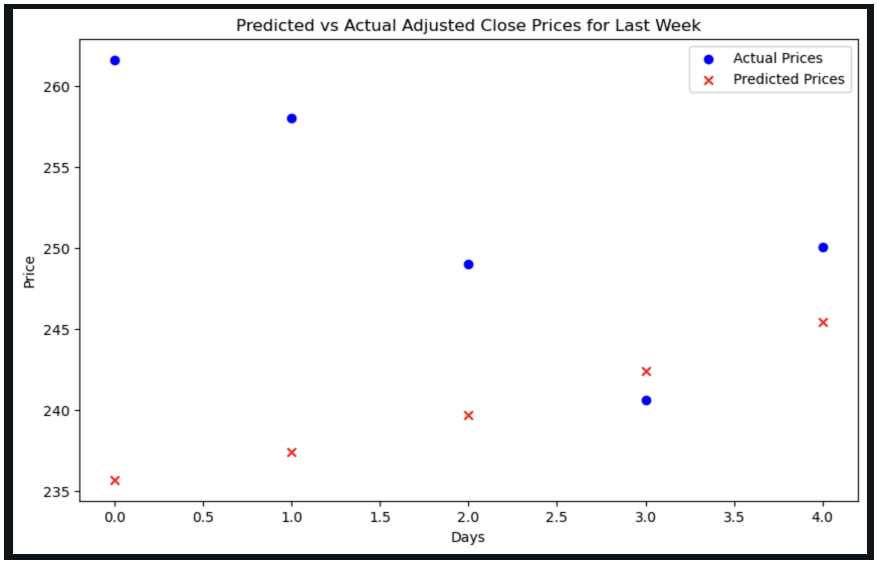
- Tweaks Made:
  - Dropout Rate: Increased dropout rates to reduce overfitting, forcing the model to generalize better by making neurons less dependent on specific patterns in the training data. (Dropout Initial: 0.2, Dropout Adjusted: 0.45)

    - Learning Rate: Adjusted learning rates to control the pace of weight updates. Experimented with lower learning rates to prevent drastic changes during training and stabilize the learning process. (Added learning rate of 0.0005)
    - 

- Incorporating More Features:
    - Attempted to add more columns from the dataset (e.g., open, high, low prices) to capture more information, but this led to increased variance in losses, further complicating the model's performance.
    - Decided to revert to using only the 'Adj Close' column due to better performance with fewer features.

3. Predictions and Current Observations:

- Ran the model to predict last week’s adjusted close prices for TSLA.
    - Results were disappointingly linear, indicating that the model isn't fully capturing the complex patterns in the time-series data and is likely over-smoothing the predictions.
    - This suggests that the model may not be learning the temporal relationships effectively or is too simplified in its architecture.



4. Next Steps:

- Address the Linearity:
    - Will investigate ways to improve the model's performance by:
        - Adding more LSTM layers or adjusting the number of neurons in existing layers.
        - Experimenting with sequence length (i.e., the number of past days used to predict future prices).
        - Further tuning of hyperparameters, such as dropout rate, batch size, and learning rate, to reduce overfitting and improve the model's complexity.
        - Exploring additional feature engineering methods to better capture price dynamics while keeping overfitting in check.