Lesson Summary

Congratulations! You have completed this lesson. At this point in the course, you know:

- Linear regression refers to using one independent variable to make a prediction.
- You can use multiple linear regression to explain the relationship between one continuous target y variable and two or more predictor x variables.
- Simple linear regression, or SLR, is a method used to understand the relationship between two variables, the predictor independent *variable x* and the target dependent *variable y*.
- Use the **regplot** and **residplot** functions in the Seaborn library to create regression and residual plots, which help you identify the strength, direction, and linearity of the relationship between your independent and dependent variables.
- When using residual plots for model evaluation, residuals should ideally have zero mean, appear evenly distributed around the x-axis, and have consistent variance. If these conditions are not met, consider adjusting your model.
- Use distribution plots for models with multiple features: Learn to construct distribution plots to compare predicted and actual values, particularly when your model includes more than one independent variable. Know that this can offer deeper insights into the accuracy of your model across different ranges of values.
- The order of the polynomials affects the fit of the model to your data. Apply Python's **polyfit** function to develop polynomial regression models that suit your specific dataset.
- To prepare your data for more accurate modeling, use feature transformation techniques, particularly using the **preprocessing** library in scikit-learn, transform your data using polynomial features, and use the modules like **StandardScaler** to normalize the data.
- Pipelines allow you to simplify how you perform transformations and predictions sequentially, and you can use pipelines in scikit-learn to streamline your modeling process.
- You can construct and train a pipeline to automate tasks such as normalization, polynomial transformation, and making predictions.
- To determine the fit of your model, you can perform sample evaluations by using the Mean Square Error (MSE), using Python’s **mean_squared_error** function from scikit-learn, and using the score method to obtain the R-squared value.
- A model with a high R-squared value close to 1 and a low MSE is generally a good fit, whereas a model with a low R-squared and a high MSE may not be useful.
- Be alert to situations where your R-squared value might be negative, which can indicate overfitting.
- When evaluating models, use visualization and numerical measures and compare different models.
- The mean square error is perhaps the most intuitive numerical measure for determining whether a