**REPORT SUMMARY**

* Implement a linear regression learner to solve this best fit problem for 1 dimensional data. Make sure your implementation can handle fits for different ”function depths” (at least to ”depth” 3) using scikit-learn and other libraries

**Output Analysis:**

* **Depth 1:**
* **MSE**: 11.6763
* At a depth of 1, the model shows a relatively high MSE, indicating a significant difference between its predictions and the actual values. This suggests an oversimplification of the model, possibly leading to underfitting. An underfit model might not capture the complexities and patterns in the data adequately.
* **Depth 2:**
* **MSE**: 0.6907
* The MSE sees a marked reduction when the model depth is increased to 2. This implies that the model, with an added layer of depth, is better equipped to capture more nuances in the data.
* **Depth 3:**
* **MSE**: 0.0204
* At a depth of 3, the MSE plunges even further, suggesting that this model achieves a close fit to the actual values. The model's predictions at this depth seem to be very accurate, given the substantially reduced error.

**Conclusion:**

* The model's performance enhances notably with increasing depth, moving from 1 to 3. The sharp decline in MSE values indicates the model's improved ability to predict more accurately with added complexity.
* Among the provided depths, the model with a depth of 3 emerges as the best performer, boasting the lowest MSE of 0.0204.
* While the depth 3 model demonstrates superior performance on this dataset, it's essential to exercise caution. Greater depth can sometimes result in overfitting, making the model excessively tailored to the training data and potentially reducing its performance on new, unseen data. To ensure the model's robustness and prevent overfitting, it's advisable to evaluate its performance on a separate validation or test dataset.

b) Implement a linear regression learner to solve this best fit problem for 1 dimensional data. Make sure your implementation can handle fits for different” function depths” (at least to ”depth” 3) in Python using only NumPy, math, matplotlib libraries. Essentially implement Linear Regression from scratch.

**Output Analysis:**

* **Degree 0:**
* **Test MSE**: 18.1306
* With a degree of 0, which is equivalent to a constant model, the MSE is quite high. This implies that the model is very simplistic and doesn't capture any trends in the data.
* **Degree 1:**
* **Test MSE**: 13.0802
* Elevating the model to degree 1, which represents a linear model, we observe a noticeable reduction in the MSE. This signifies that a linear model provides a better fit to the data compared to the constant model.
* **Degree 2:**
* **Test MSE**: 13.7251
* Interestingly, the MSE increases slightly when moving to a quadratic model (degree 2). This might suggest that adding a quadratic term doesn't necessarily enhance the fit for the given dataset and might even introduce some unnecessary complexity.
* **Degree 3:**
* **Test MSE**: 2.3516
* At degree 3, which represents a cubic model, there's a substantial decline in the MSE. This denotes that the model is significantly better at fitting the data and capturing its intricacies at this degree.

**Conclusion:**

* Progressing from degree 0 to 3, the model's performance sees the most remarkable enhancement at degree 3. The MSE at this degree is significantly lower than the preceding degrees, suggesting a more accurate fit to the data.
* The best-performing model is with degree 3, registering an MSE of 2.3516.
* It's noteworthy that while a cubic model (degree 3) outperforms the other models on this dataset, one must remain vigilant about potential overfitting. Higher-degree polynomials can sometimes fit noise or random variations in the training data, which might not generalize well to new data.

**Graph Analysis:**

**1.**



**Observations:**

* **Data Points:** The red dots represent the data points in the graph. These data points display a clear nonlinear pattern, resembling a wave or a cyclical trend. There are distinct peaks and troughs observable, especially around x-values of -2, 0, and 2.
* **Degree 1 Fit:** The blue line represents a linear regression fit, which is evident from its straight nature. This fit is consistent with a polynomial of degree 1.

**Analysis:**

* The linear fit (degree 1) does not capture the underlying trend of the data points effectively. Given the nonlinear nature of the data points, a linear model is too simplistic to represent the intricacies of the data accurately.
* The portions of the data around x-values of -2 and 2 show a stark deviation from the linear fit. The linear model fails to capture the peaks and troughs of the data, suggesting that it's not the optimal choice for this dataset.
* Even though the linear fit might have reduced the MSE from the degree 0 model (as per the provided numerical data), it's evident visually that the model is inadequate for capturing the overall structure of the data.

**2.**



**Observations:**

* **Data Points:** As before, the red dots on the graph represent the data points. The dataset showcases a clear nonlinear, cyclical pattern, with evident peaks and troughs.
* **Degree 2 Fit:** The blue curve represents the fit of a polynomial regression model of degree 2, which is evident from its parabolic nature.

**Analysis:**

* The quadratic fit (degree 2) seems to capture the trend of the data points in the left half of the graph (around x-values of -3 to 0) relatively well, especially the trough around x=-2.
* However, the right half of the graph (around x-values of 0 to 3) indicates that the degree 2 fit struggles to adapt to the wave-like pattern of the data points. Specifically, the model does not capture the peak around x=2, causing a significant deviation between the data points and the fitted curve.
* Compared to the degree 1 linear fit, the quadratic model does a better job in certain regions, especially around the trough. However, it's still insufficient in representing the entire complexity of the dataset.

**3.**

**Observations:**

* **Data Points:** The red dots continue to represent the data points, displaying a distinct nonlinear, cyclical pattern.
* **Degree 3 Fit:** The blue curve signifies the polynomial regression fit of degree 3. It showcases a more intricate curve, adapting to the wave-like structure of the data points.

**Analysis:**

* The degree 3 fit remarkably captures the overall trend of the data points, providing a detailed representation of the dataset's wave-like structure.
* The model successfully captures the trough on the left side (around x = -2) and the peak on the right side (around x = 2), showing its ability to understand and represent the intricate oscillations of the data.
* There are minimal deviations between the data points and the curve, indicating a close fit. This aligns well with the provided numerical data, which stated that the degree 3 polynomial model had the lowest MSE.
* Compared to the degree 1 and degree 2 fits, the degree 3 model is clearly superior in capturing the nuances of this dataset.

**c) Apply your regression learner to the data set that was generated for Question 1b) and plot the resulting function for “function depth” 0, 1, 2, and 3. Plot the resulting function together with the data points. Results must include a plot.**

**1.**

**Observations:**

* **Data Points:** The red dots on the graph represent the given data points. These points show a clear nonlinear pattern, demonstrating multiple oscillations.
* **Degree 1 Fit:** The blue straight line is the representation of a polynomial regression model of degree 1, essentially a linear fit to the data.

**Analysis:**

* The linear (degree 1) fit is a simple straight line, and from the graph, it's evident that it doesn't capture the nonlinear nature of the data.
* On the left side of the graph, especially around x-values of -3 to -1, the line does not capture the trough in the data, leading to significant deviations.
* Similarly, on the right side (around x-values of 1 to 3), the line fails to capture the peak in the data points, causing further deviations.
* Overall, the straight line indicates an upward trend in the data, but this is an oversimplification given the wave-like structure of the data points.

**2.**

**Observations:**

* **Data Points:** The red dots represent the observed data points. These data points appear to follow a more complex, wave-like structure.
* **Degree 2 Fit:** The blue curve is the representation of a polynomial regression model of degree 2.

**Analysis:**

* The polynomial regression of degree 2 presents a parabolic shape. It manages to capture the trough in the data around x-values of -3 to -1 more accurately than the degree 1 fit.
* However, as the x-values move towards positive territory, especially from 1 to 3, the degree 2 polynomial model fails to capture the peak in the data. The curve maintains its upward trajectory without adjusting to the data's oscillating pattern.
* This degree 2 fit represents an improvement over the degree 1 fit for the left half of the data but still fails to fully capture the intricacies of the data's nonlinear nature.

**3.**

**Observations:**

* **Data Points:** The red dots represent the observed data points. These points continue to demonstrate a wave-like pattern.
* **Degree 3 Fit:** The blue curve showcases the polynomial regression model of degree 3.

**Analysis:**

* The degree 3 polynomial regression curve has an S-shaped structure, which allows it to capture both the trough on the left side and the peak on the right side of the graph.
* This curve appears to align closely with the data, especially around the x-values of -3 to -1 and 1 to 3, capturing the data's oscillatory nature.
* There are some discrepancies in the middle region (around x=0), where the curve does not perfectly align with the data points. However, this misalignment is minimal compared to the fits of lower degrees.

**d) Evaluate your regression functions by computing the error on the test data points that were generated for Question 1c) Which” function depth” would you consider the best prediction function and why? With which values of d do you get minimum error? Results must include a plot.**

**Graph Analysis:**

**Observations:**

* **Mean Squared Error (MSE):** The blue line with markers represents the MSE for polynomial regression models of varying degrees, starting from degree 1 to degree 9.
* **Min Error at Degree 9:** The red dashed line indicates the degree of polynomial that achieved the minimum MSE.

**Analysis:**

* **Degrees 1-3:** We observe a drastic decrease in MSE from degree 1 to degree 2 and a further decline to degree 3. This suggests that higher polynomial degrees initially provide a better fit to the data.
* **Degrees 3-9:** Post-degree 3, the decline in MSE becomes more gradual. The MSE stabilizes and decreases slightly as the polynomial degree increases.
* **Optimal Degree:** The chart indicates that the polynomial of degree 9 has the lowest MSE among all the degrees tested. However, the reduction in error from degree 4 to degree 9 is relatively small.

**e) Clearly briefly explain the code workflow of part b)**

* **Utility Functions**:
* **load\_data(train\_file, test\_file)**: Loads the training and test data from the provided files.
* **prepare\_features(x, depth)**: Prepares the polynomial features for the data up to the specified depth (degree).
* **linear\_regression(X, y)**: Computes the regression coefficients using the closed-form solution.
* **predict(X, theta)**: Predicts the output given the input features and regression coefficients.
* **plot\_results(x, y, theta, depth)**: Plots the original data points and the polynomial fit.
* **Main Workflow**:
* The data is loaded using the **load\_data** function.
* For each polynomial degree (from 0 to **max\_depth**), the following steps are taken:
* Polynomial features are prepared using **prepare\_features**.
* The polynomial regression model is trained using the **linear\_regression** function.
* Predictions are made on the test set using the trained model.
* The Mean Squared Error (MSE) for the test set is calculated and printed.
* The training data and polynomial fit are visualized using the **plot\_results** function.
* After evaluating all polynomial degrees, the best degree (with the lowest MSE) is identified and printed.
* **Execution**:
* The main workflow is executed when the script is run using the **if \_\_name\_\_ == "\_\_main\_\_":** construct.

**The code performs the following**:

* Loads training and test datasets.
* For each polynomial degree:
* Trains a polynomial regression model.
* Tests the model on the test data and calculates its MSE.
* Plots the training data and the model's fit.
* Identifies and reports the best polynomial degree based on the test MSE.