Mini Project III -MAL 7080- Mathematical Modeling and Simulations

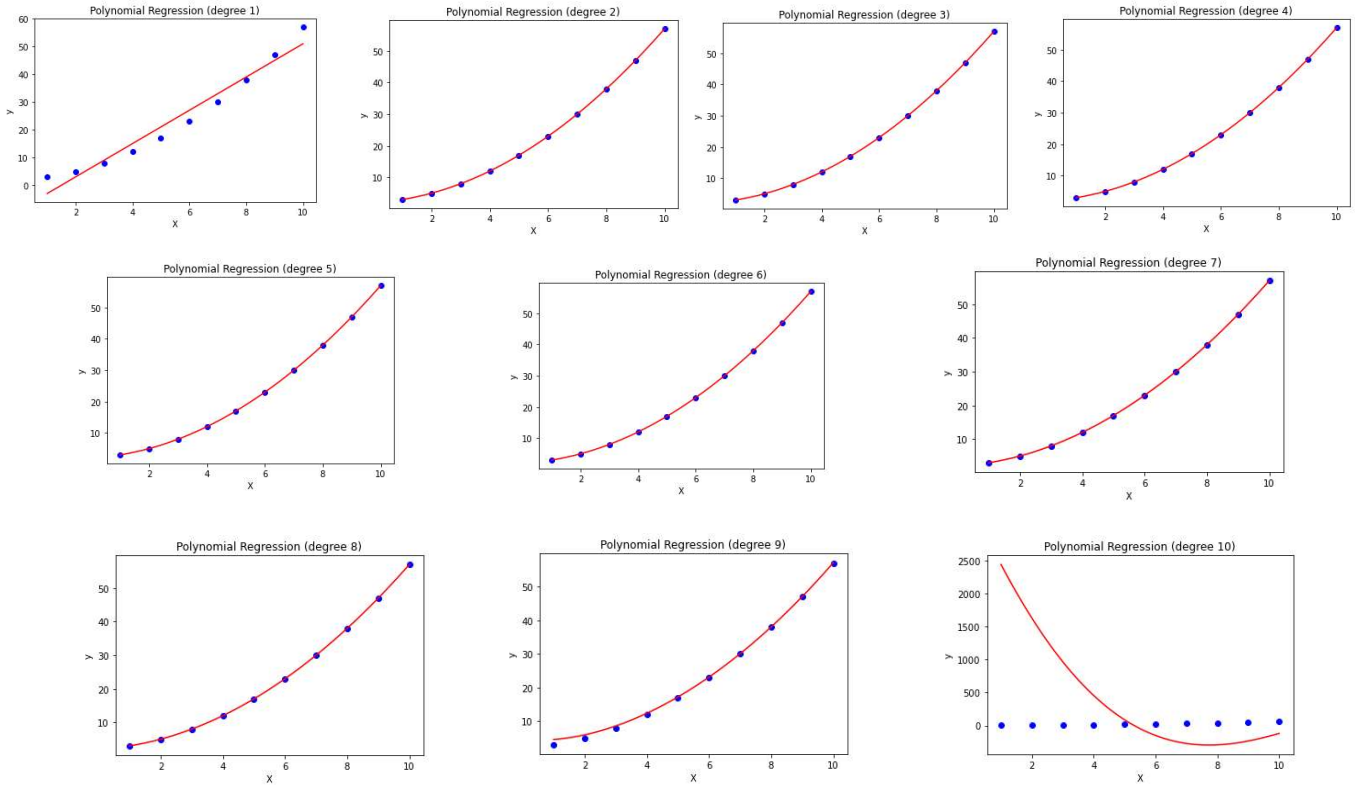
Rakesh Reddy Kadapala – M22AI608

1q>Construct a class to perform nth degree polynomial regression. You are provided with the basic class structure to start with and are required to complete the methods within the class. Fit a curve for any dataset of your choice. Run this regression analysis for degrees 1 to 10, then report the best fitting curve by analysing the error. Make sure to plot the fitting curves as well.
NOTE: The dataset should not be linear or be generated using any internal functions.

Polynomial Regression:
In the code class `PolynomialRegression` for performing nth degree polynomial regression. The class contains three methods: `__init__`, `fit`, `predict`, and `plot_fit`.

I. `__init__` method initializes the degree of the polynomial curve that will be fit to the data.
II. `fit` method takes X and y as input and fits an nth degree polynomial curve to the data using the normal equation formula.
III. The `predict` method takes input X and returns the predicted values of y.
IV. The `plot_fit` method takes input X and y, plots the scatter plot of the data points, and overlays the predicted curve on the scatter plot.

The code then creates some random dataset, specifies a range of degrees from 1 to 10, and fits polynomial curves for each degree to the data using the `PolynomialRegression` class. then calculates the root-mean-square error for each degree of the polynomial curve and stores them in the `errors` list. The code then selects the degree of the polynomial curve with the lowest error and prints it as the best fitting curve.



The `plot_fit` method provides a clear visualization of the data points and the predicted curve.

Best Fitting Curve is: Degree 2

2q> Make a file called multiple regression.py and implement multiple linear regression using stochastic gradient descent technique from scratch. You can use any dataset of your choice, given that its features are more than 3. You can implement it according to you as there is no boilerplate code provided for this task.

Multiple Linear Regression:

The class `MultipleLinearRegression` takes two parameters: the learning rate and number of iterations. The `fit` method takes in the input `X` and target `y` as numpy arrays, initializes the parameters theta to zeros, and concatenates a column of ones to the input data to account for the intercept.

The method then updates theta using stochastic gradient descent, randomly selecting one data point at a time and computing the gradient and updating theta until the number of iterations is reached.

The `predict` method takes in new data `X` and returns the predicted target `y` using the learned theta values.

To test the implementation, a random dataset with 4 features and 1 target is created using numpy. The true relationship between the features and target is defined as a linear combination with random weights and some added noise. The model is then created and trained on the dataset with a learning rate of 0.1 and 10,000 iterations.

The model is used to predict the target on new data and the predictions are printed. The Predicted values are:

```
   ...: y_pred
Out[1]:
array([[ 9.91982031],
       [ 8.60746702],
       [ 7.60215476],
       [13.84957284],
       [10.96818129],
       [ 6.18009979],
       [10.74882132],
       [10.18515663],
       [12.91236179],
       [ 4.87362329]])
```