CS 5805: Machine Learning I
Spring 2025
Homework 2
Date Assigned: 2/5/2025
Date Due: 2/17/2025

1. (15 points) Consider the function $y = 3x^3 + 4x^2 + 5x + 6$. Let us try to learn this function using linear regression via gradient descent. In other words, the linear regressor is learning values of (a,b,c,d) in $y=ax^3 + bx^2 + cx + d$.

   Let us create a 4-example dataset to train the regressor. Use x-values of -1, 0, 1, and 2 to find corresponding values of y and constitute your dataset. Let us begin with initial settings of a=1,b=1,c=1,d=1.

   Calculate and submit the results of the next three iterations of gradient descent with a learning rate of 0.01.
   Will your algorithm eventually converge to the correct values? Why/why not?
   In general, will your algorithm converge to the correct values irrespective of the starting point and irrespective of the examples? Note that the function is cubic.

2. (15 points) Assume we wish to learn $y = ax^2 + bx + c$ via linear regression from a given dataset. But you are given only three regression learning algorithms as follows:
   - A1: Learns a function of the form $y=ax^2$
   - A2: Learns a function of the form $y=ax$
   - A3: Learns a function of the form $y=a$

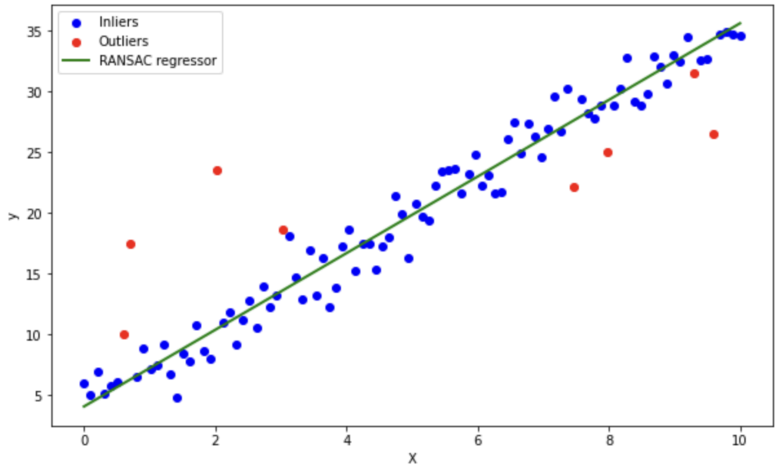
   If these are the only three algorithms given (and no other algorithms are available or can be used), is there a way to learn the original function $y = ax^2 + bx + c$ by some combination of the given three algorithms? If the answer is yes, explain the procedure (and make sure it works for all datasets). If the answer is no, explain why not.

3. (20 points) A problem with linear regression, specifically the mean squared error criterion (MSE), is that it is sensitive to outliers. Due to squaring, MSE penalizes large errors more heavily than small errors and this makes it sensitive to outliers. One way to deal with outliers in regression is the so-called RANSAC (RANdom SAmple Consensus) regressor, available in scikit-learn as the RANSACRegressor. It aims to fit a linear model while ignoring data points that deviate significantly from the estimated model. Here is (approximately) how this algorithm works:

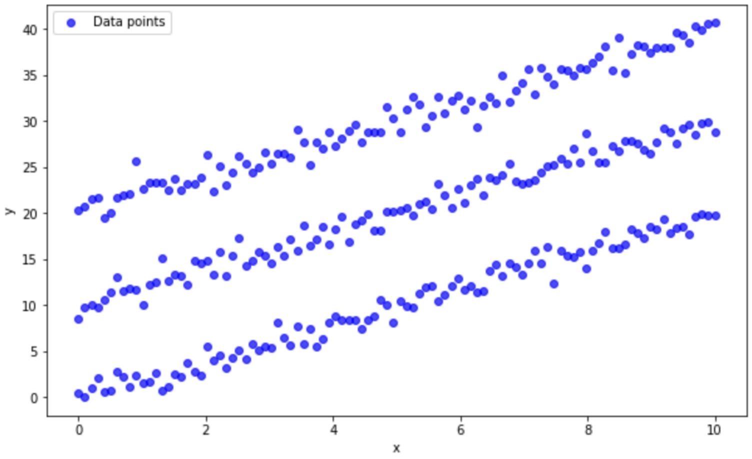
   a. The algorithm repeatedly selects a random subset of the data.
   b. Using this randomly chosen subset, RANSAC fits a linear regression model.
   c. The fitted model is then used to predict values for all points in the dataset.
   d. The error (typically the absolute or squared residual) for each data point is computed.
   e. Points whose errors are below a predefined threshold (called the residual threshold) are called **inliers**—they are consistent with the candidate model.
   f. The algorithm counts how many data points are inliers.

g. If the number of inliers exceeds a pre-set threshold (or is the best seen so far), the candidate model is considered the current best.
h. Steps a–g are repeated for a fixed number of iterations (or until a certain confidence level is reached).
i. The model with the highest number of inliers (or the best quality as measured by some metric) is selected.
j. Optionally, once the best set of inliers is identified, the model is re-fitted using all the inliers to obtain a final, refined estimate.

See attached some code with this assignment (RANSAC.ipynb) and below a plot of how it works. Note that the regression has learnt to flag the red points as outliers and inferred the regression from only the points deemed as inliners.

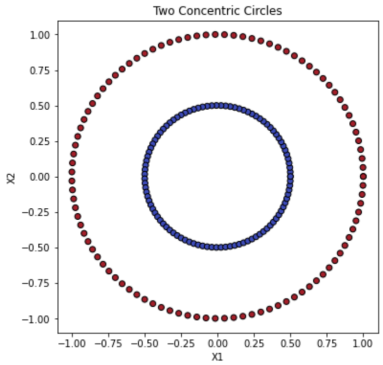


Your goal is to see if you can make RANSAC work for the below dataset (which has three line streaks). Code to generate this dataset is also supplied with this assignment (threelinestreaks.ipynb). Can you make the linear regression model learn one of the streaks (the choice of which one to learn is yours to make) and treat the other two lines as outliers?



For full credit, submit your notebook and observations on if (and how, or why not) you were able to make RANSAC work for this dataset.

4. (20 points) Consider the two concentric circles dataset (notebook to generate it supplied with the assignment - 2concentriccircles.ipynb). It looks like this:



Explain how a logistic regression classifier can be made to separate these two classes (each circle denoting one class). For full credit, explain what you did, your notebook, and an assessment of your classifier's performance.

5. (30 points) Consider the Stars Classification dataset available at (https://github.com/YBIFoundation/Dataset/raw/main/Stars.csv) and build a logistic regression classifier to predict the spectral class out of seven classes (O,B,A,F,G,K,M) [fun reading at https://en.wikipedia.org/wiki/Stellar_classification]. Begin by loading the data and performing exploratory analysis.

Conduct two experiments. For each experiment, conduct cross validation, and choose the number of folds by yourself. Rationalize why this is a good setting.
5.1. Using only 4 features as input X, ['Temperature (K)', 'Luminosity (L/Lo)', 'Radius (R/Ro)', 'Absolute magnitude (Mv)'], predict y of ['Spectral Class'] with 7 classes.
5.2. Using the same 4 features X, group the Spectral Class into 3 classes (Hot stars or Blue-White Stars (O, B, A), Intermediate stars or Yellow Stars (F, G), Cool stars or Red-Orange Stars (K, M)), and predict which of the 3 classes your instance falls in.

For each experiment, after fitting the logistic regression model, evaluate its performance using various classification metrics such as accuracy, precision, recall, F1 score, and the confusion matrix. Discuss what each metric measures, their relevance to classification, and any limitations they might have in this context. Additionally, plot the Receiver Operating Characteristic (ROC) curve and compute the Area Under the Curve (AUC). Explain what the ROC curve and AUC reveal about your models' abilities to distinguish between stars.

Finally, examine the coefficients of your logistic regression models. Give a very precise technical interpretation of the coefficients of the model. Compare the coefficients across the two settings and discuss any differences you observe.