Assignment 3

1. Create XOR dataset using the following link: https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpc_xor.html
    1. Then, write a class for Binary Logistic regression (from scratch). **[0.5 marks for miscellaneous class interface creation]**
    2. You can use JAX, PyTorch or Tensorflow for automatic gradient computation. The number of iterations should be an argument while fitting. **[0.5 marks for correctly using autograd]**
    3. You should use cross-entropy loss (CE loss), additionally allow for a L2 penalty term and have a corresponding coefficient corresponding to L2 penalty. Thus, overall loss = CE loss + Lambda X L2 norm of theta vector. The Lambda would default to 0. and should be an argument while initialisation. **[1 mark for defining the loss function correctly]**
    4. Randomly split train/test as 60:40
    5. Draw the decision surface as shown in the sklearn document and also show the train and test accuracy. **[0.5 marks]**
    6. Draw the loss v/s iterations for lambda = 0 and 0.5 **[.5 marks]**
    7. Report train and test accuracy and explain your result. Which train points and test points are misclassified? Why? Use decision surface from above to help answer **[1 mark]**
2. Solve the above but instead of using your code from scratch, use the neural network (NN)interface in Flax or PyTorch or Tensorflow and compare the results with yours. (Logistic regression is the simplest NN) **[1 mark]**
3. Use autogluon to compare the performance of different classifiers on the above dataset. **[1 mark]**
4. Use the IRIS dataset and create your custom (from scratch) implementation for Multi-class Logistic regression. [0.5 marks for miscellaneous class interface creation]. If you want you can have a single implementation that handles both binary as well as multi-class classification (and marks will be accordingly adjusted)
    1. Split train:test at 80:20 but use sklearn StratifiedKFold instead of regular KFold.
    2. Draw the decision surface **[0.5 marks]**
    3. You should use cross-entropy loss (CE loss), additionally allow for a L2 penalty term and have a corresponding coefficient corresponding to L2 penalty. Thus, overall loss = CE loss + Lambda X L2 norm of theta vector. The Lambda would default to 0. and should be an argument while initialisation. **[1 mark for defining the loss function correctly]**
    4. Draw the loss v/s iterations for lambda = 0 and 0.5 **[0.5 marks]**
    5. Report train and test accuracy and explain your result. Which train points and test points are misclassified? Why? Use decision surface from above to help answer **[1 mark]**

6. Solve the above but instead of using your code from scratch, use the neural network interface in Flax or PyTorch or Tensorflow and compare the results with yours. **[1 mark]**