Exercise 2.1

With regression common loss functions are MSE (Mean Squared Error) and MAE (Mean Absolute Error). Huber Loss (or Smooth Mean Absolute Error) is the combination of MSE and MAE. It approaches MSE when the error is small and MAE when the error is big. The limit for the small/big error is a hyperparameter δ. The Huber Loss function for one sample is the following

$$\mathcal{L}(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{when } |y_i - \hat{y}_i| \leq \delta \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

The Huber Loss is less sensitive to outliers than MSE. It is also continuous and differentiable which makes it easier to use in optimization algorithms using gradients. The choice of the δ value is critical because it determines what you're willing to consider an outlier.

Given the following results of a certain model, calculate the loss values for Huber Loss with three different δ values: 1.0, 2.0 and 3.0.

y	$\hat{y}$
2	1.8
1.2	1.6
1	0.4
2	1.9
1.6	1.7
0.9	0.6
4	1.2
6	1.4

Which is the value of δ that gives the smallest average loss?

What is the value of the largest average loss? Give the answer rounded to three decimals.

Exercise 2.2

In Scikit-Learn (module `model_selection`) there are a class called `GridSearchCV`.This class can be used to tune hyperparameters of a machine learning model.

Do the following hyperparameter tuning using `GridSearchCV`:

- Use the Iris dataset loaded with `sklearn.datasets.load_iris`.
- Split the data to train and test data using `sklearn.model_selection.train_test_split`.
  - Split ratio is 80/20, i.e. 80% for training data and 20% for test data.
  - Use a seed value 425, i.e. parameter `random_state`. This is important only for getting the same right answer.
- Make a Support Vector Machine model as follows: `sklearn.svm.SVC()`
- Tune the following hyperparameters:
  - Parameter `C` values 0.1, 1, 10 and 100.
  - Parameter `gamma` values 1, 0.1, 0.01 and 0.001.
  - Parameter `kernel`values 'linear', 'poly' and 'rbf'.
  - So the total number of models is 48.
- Use cross-validation with 4 folds in `GridSearchCV`.
- Use `'accuracy'` as a scoring value for the models to be trained.

What are the best values for parameters?

C:

gamma:

kernel:

Exercise 2.3

Regularization is used in machine learning to prevent overfitting. In this exercise you'll explore the concept of regularization using a simple linear regression model.

Use the data found in here, i.e. read the csv file with a pandas (`pandas.read_csv`) to make a DataFrame:
https://raw.githubusercontent.com/haniemi/deeplearning/main/data/regression_gen_data.csv

DataFrame contains three columns X1, X2 and y. Independent variables are $X = (X1, X2)$ and the predicted (dependent) variable is y.

Split data into training and testing sets using ratio 80/20 and random_state value 42.

Implement models using the following classes from `sklearn.linear_model` library: `LinearRegression`, `Ridge`, `Lasso` and `ElasticNet`.

For the `Ridge` and `Lasso` find the optimal value for the regularization parameter (= hyperparameter) `alpha` using values 0.01 - 0.99 with step size 0.01, i.e. values 0.01, 0.02, ..., 0.99. Use the scikit-learn object `GridSearchCV` with parameters `cv=5`, `scoring='neg_mean_squared_error'` to find the optimal `alpha`.

For the `ElasticNet` find the optimal value for the regularization parameters `alpha` and `l1_ratio`. For `alpha` use the same values as above and for `l1_ratio` use values 0.1- 0.9 with step size 0.1, i.e. values 0.1, 0.2, ..., 0.9.

Evaluate the performance of all four models using test data. Use the function `sklearn.metrics.mean_squared_error` to evaluate the model performance.

What is the optimal value for `alpha` in Ridge regression?

Which model produces the biggest loss value for the test data?