Exercise Sheet 5

1. Exercise: k-Nearest Neighbors Classification

(a) Write a Python class to implement a k-Nearest Neighbors (k-NN) classifier. The class should be called `KNeighborsClassifier` and should include the following methods:

   (i) `__init__`: This should initialize the k-NN classifier with the number of neighbors to consider and the distance metric to use.

   (ii) `fit`: This should fit the model to the provided training data.

   (iii) `predict`: This should predict class labels for the provided test data.

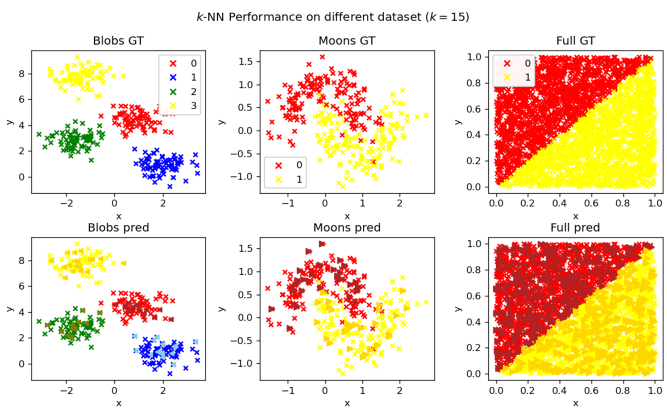
(b) The function of k-NN classifiers is heavily dependent on the specification of a distance metric. For this exercise, the Euclidean distance metric will be used. Implement function `euclidean_distance(xi, X)` that computes the Euclidean distances between between a point $\boldsymbol{x}_i$ and all data points $\boldsymbol{X}$.

(c) Write a Python function called `evaluate(y_pred, y_gt)` to determine the accuracy of your classification model's predictions. Calculate the accuracy as the proportion of correct predictions over the total number of predictions.

(d) Load the data sets `data_blobs.pkl`, `data_moons.pkl` and `data_full.pkl`. You can find the files in the Moodle course.

(e) Initiate three k-NN classifiers from (a) with $k = 1$, $k = 15$ and $k = 30$. Each classifier applies Euclidean distance metric. Train the k-NN classifiers using the train data and then use it to predict the labels for the test samples. Calculate the accuracy of these predictions.

(f) Generate the following plot based on the prediction of the k-NN classifier ($k = 15$). The first row shows the ground truth data for each dataset. The second row shows the ground truth for the train data (marker 'x') and the predicted labels (marker '>') for the test data.



2. Exercise: Nadaraya-Watson estimator

The Nadaraya-Watson estimator is a non-parametric regression technique defined as

$$f_{\mathrm{NW}}(\boldsymbol{x}) = \sum_{m=1}^{M} \underbrace{\left(\frac{\kappa\left(\boldsymbol{x} - \boldsymbol{x}_m\right)}{\sum_{n=1}^{M} \kappa\left(\boldsymbol{x} - \boldsymbol{x}_n\right)} \right)}_{w_m} \boldsymbol{y}_m. \tag{1}$$

Based on a set of data points $\boldsymbol{x}$, it calculates the contribution of each point based on its proximity to the point under consideration $\boldsymbol{x}_m$, using a kernel function $\kappa()$. The closer points, therefore, have more influence on the estimation than points further away. Follow the subsequent steps to implement a Nadaraya-Watson estimator without using built-in functions.

(a) For this exercise, a Gaussian kernel function is applied to the Nadaraya-Watson estimator. Implementation the kernel function `def gkernel(d_m, h)` which takes the bandwidth h and the distance value $\boldsymbol{d}_m$ such that

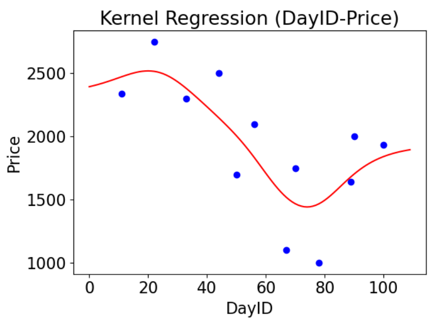
$$\kappa(\boldsymbol{d}_m, h) = \frac{1}{h\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\boldsymbol{d}_m}{h}\right)^2}, \quad \text{where } \boldsymbol{d}_m = \boldsymbol{x} - \boldsymbol{x}_m. \tag{2}$$

(b) Implement the function `def weight_m(K)` that returns the weights w_m according to Eq. 1.

(c) Use the `pandas` library to represent the following tabular data.

DayID	11	22	33	44	50	56	67	70	78	89	90	100
Price	2337	2750	2301	2500	1700	2100	1100	1750	1000	1642	2000	1932

(d) Implement the Nadaraya-Watson estimator using the functions of (a) and (b). Apply this on the DayID-Price data set. Your task is to extract the price information for the days ranging from 1 to 110. Set your bandwidth parameter to $h = 10$. Once completed, create a plot to visualize your result and the data points as illustrated below.



(e) How does changing the bandwidth parameter h impact the regression result?