Machine learning for Neuroscience (Tel Aviv University, 2020)

Exercise 1

Use the following code snippet to load the training set and test set

```
import pandas as pd
dataset_dir =
"https://raw.githubusercontent.com/probml/pmtkdata/master/knnClassify3c/"
train_dataset_path = dataset_dir + "knnClassify3cTrain.txt"
test_dataset_path = dataset_dir + "knnClassify3cTest.txt"

# Train dataset
train_dataset = pd.read_csv(train_dataset_path,
                            names=["x1", "x2", "class"],
                            delimiter=" ")
X_train = train_dataset.iloc[:, :-1].values
y_train = train_dataset.iloc[:, -1].values

# Test dataset
test_dataset = pd.read_csv(train_dataset_path,
                           names=["x1", "x2", "class"],
                           delimiter=" ")
X_test = test_dataset.iloc[:, :-1].values
y_test = test_dataset.iloc[:, -1].values
```

1. (10 points) plot a pairs plot and a box-whisker plot based using the training set.
2. (10 points) Based on visual inspection of the pairs plot, write a function that classifies based on rules e.g.,

   ```
   class classificationRules:
     def predict(x_1,x_2):
       if (x_1>7 and x_2<9):
         my_class=1
       elif x_2<4:
         my_class=2
       else:
         my_class=3
       return my_class
   ```

3. (10 points) Apply the rule based classifier in (2) to predict the the test set and report the misclassification rate.
4. (20 points) Apply *k*-NN (use this function) on the the test set and plot the miscalssification rate for k=1, k=5 and k=10 (plot misclassification rate v.s. $1/k$).
5. (20 points) Apply the k-NN naïve implementation here on the test set (with k=1) and calculate the misclassification rate.
6. We can write the *k*-NN algorithm as follows. Denote the k observations which are nearest to the query point $\boldsymbol{x}^*$ by $\mathcal{N}_k(\boldsymbol{x}^*)$. The prediction $\hat{Y}(\boldsymbol{x}^*)$ is the most common value among the k training observations nearest to $\boldsymbol{x}^*$. This *majority vote* strategy can be expressed by

   $$\hat{Y}(\boldsymbol{x}^*) = \underset{g \in \mathcal{G}}{argmax} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{N}_k(\boldsymbol{x}^*)} \mathbb{I}_{\{y_i = g\}}.$$

   In the above, g represent a class from the set of possible classes $\mathcal{G}$ and $\mathbb{I}_{\{y_i=g\}}$ is an indicator function

   $$\mathbb{I}_{\{y_i=g\}} = \begin{cases} 1 & \text{if } y_i = g \\ 0 & \text{else.} \end{cases}$$

   In a *distance weighted k-NN* , the contribution of each member of the neighbourhood of the query point $\boldsymbol{x}^*$ is weighted according to the distance to the query point

   $$\hat{Y}(\boldsymbol{x}^*) = \underset{g \in \mathcal{G}}{argmax} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{N}_k(\boldsymbol{x}^*)} w_i \mathbb{I}_{\{y_i = g\}}$$

   where the weight is inversley proportional to the squared distance

   $$w_i = \begin{cases} \frac{1}{d(\boldsymbol{x}_i, \boldsymbol{x}^*)^2} & \text{if } \boldsymbol{x}_i \neq \boldsymbol{x}^* \\ 1 & \text{else.} \end{cases}$$

   In the above $d(\boldsymbol{x}_i, \boldsymbol{x}^*)$ represents a distance function.

   (15 points) Modify the code in (5) so that it implements a *distance weighted k-NN*.
7. (10 points) Repeat (4) by using the modified code from (6).
8. (5 points) What would you do if the features are measured by differetn units?