=> Machine Learning Assignment 3 <=
150 MARKS

This is a major practical with a boss battle!

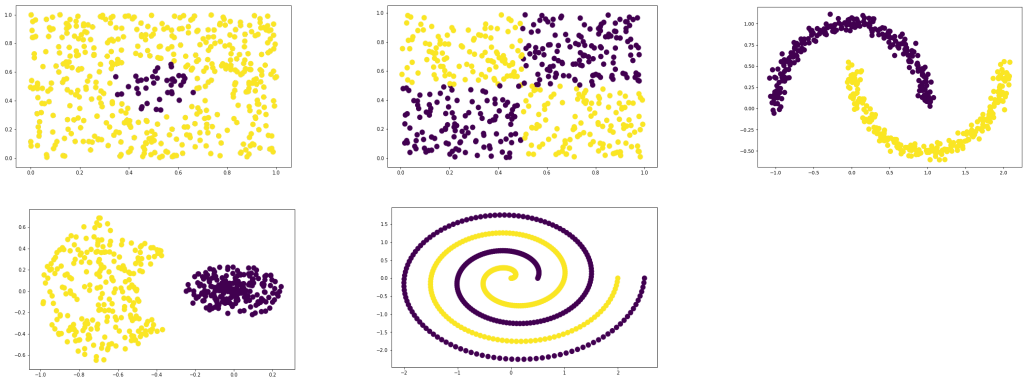
Deadline

Submit all tasks on RuConnected by **the deadline.**

Task 1: *Best* SVM Kernel? [4 + 4 + 4 + 4 + 4 = 20 Marks]

The code provided in SVM_Exercises.py includes generated datasets for classical mathematic problems with visualization code. Complete the code using the SVM kernel which will perform best in terms of accuracy on the data given in the code. You may need to fix some errors and/or transform the data,

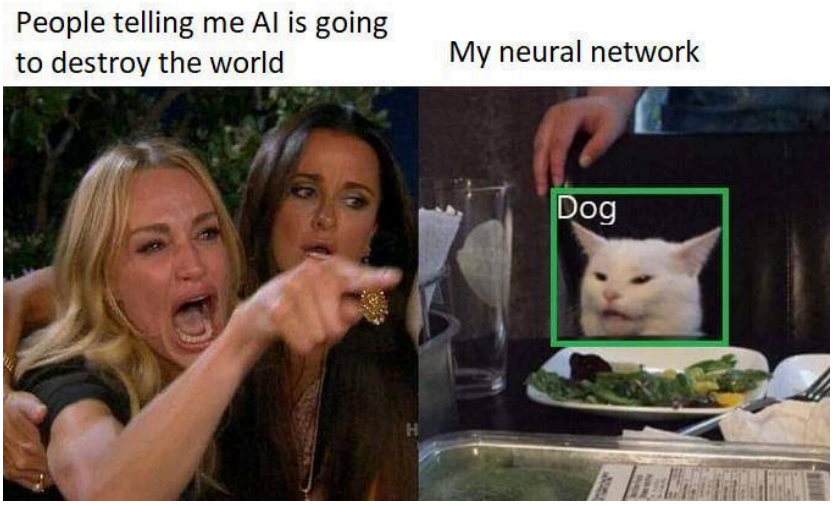
The result per dataset should print the **accuracy score** of the SVM model and **visualize** the **optimal hyperplane** per dataset. **Hint**: you are aiming for the highest accuracy on all data, so overfitting is allowed (do not split this data).



Deliverables:

- Modified Python Script, with print statements to output the results

Task 2: Cats vs. Dogs with SVMs [0 + 5 + 10 + 3 + 2 + 10 = 30 marks]

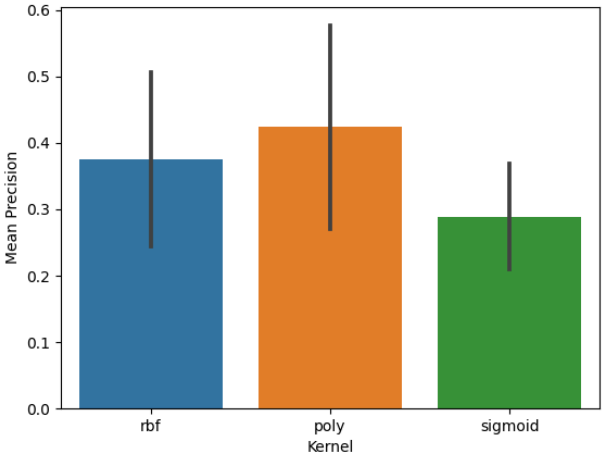


One day, the Singularity will eventually result in the AI takeover, and SkyNet will rule us all. But, before building our robot overlords, we must be happy with the example of getting them to know the difference between a cat and a dog. Small steps.

SVM_MLP.py imports a small dataset consisting of cats and dogs. It includes an in-built method that, when given the main directory of the cats and dogs dataset, will import them exactly like a regular Scikit dataset! Use this script as a starting point to do the following:

(Hint! You can find a nice example of Grid Search, amongst other things, in 9_SVM_Grid_Search_CV.py or the zip file of the lectures)

1. Split the training and test set to 50:50.
2. Scale appropriately
3. Fit SVM models to allow for parameter estimation using grid search with 10-fold cross-validation. The **objective** of the grid search is the **highest weighted-precision score**. Use the following hyperparameters to guide your search:
    - C: 0.01, 1, 10
    - gamma: <you decide>
    - kernel: ‘rbf’, ‘poly’, ‘sigmoid’
    - **Refine your parameters after the first run**
4. Grid results should be formatted similarly, but repeated for each set of hyperparameters:
    - `0.255 (+/-0.007) for {'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}`
    - `0.559 (+/-0.025) for {'C': 0.1, 'gamma': 0.01, 'kernel': 'poly'}`
    - `0.255 (+/-0.007) for {'C': 0.1, 'gamma': 0.01, 'kernel': 'sigmoid'}`
    - `0.255 (+/-0.007) for {'C': 0.1, 'gamma': 0.001, 'kernel': 'rbf'}`
5. Include a classification report (obviously on the test set)
6. Visualize the Grid results per kernel using a bar-whisker graph or your preferred equivalent. I.e. Include the mean and standard deviation of the best folds per kernel. Your bar graph layout should appear similar to this:



Deliverables:

- Python script with relevant print statements

Task 3: Cats vs. Dogs with MLPs [20 marks]

The SVM model was not good enough! We need better feature processing for a better model.

Use everything you've learned up to now and additional research to attempt to outperform the **test weighted precision** of the optimal SVM algorithm by building an optimised MLP model.

- The more effort you put into your feature processing, the higher the mark you get.
- *You must put the SVM vs MLP best Grid results on a single bar-whisker graph*

Deliverables:

- Modified Python script, which nicely displays the results of all classifiers

Task 4: Boss battle [80 Marks]

We need better models! Let's write a script that can compare a whole bunch of models! Give yourself a mark out of 80 at the top of your program (where your name should be)! To get 50% of the marks:

1. The user must specify a minimum of four command-line arguments:
    - --dataset: Choices for breast cancer, diabetes, *digits*, kddcup99, *iris*, or *wine* *or a CSV*
    - --classifiers: List classifier(s) as comma-separated arguments
    - --eda: by default, keep it simple please
    - --metric: for comparison
    - All arguments should obviously be documented using something like --help
    - **Provide example commands at the top of your program so that I don't pull out my hair:**
    - `python Task4.py --dataset iris --classifier dt --eda scatter --metric accuracy`
    - `python Task4.py --dataset iris --classifier dt rf svm --eda pairplot`
2. Complete the script. The script should be able to do classification on the specified dataset using the specified classification algorithm(s):

        - k-Nearest Neighbours
        - Logistic Regression
        - Decision Tree
        - Random Forest
        - AdaBoost and/or GradientBoost
        - Support Vector Machine
        - Multilayer Perceptron
3. At least include the following based on **Best Practices:**
        - 0.7/0.3 Train/Test split
        - Exploratory Data Analysis argument, e.g. VFM, pairplot, swarmplot etc.
        - Feature processing such as scaling and selection/extraction, etc.
        - Pipeline to combine all processes
        - Score the **training** set of a classifier using **n**-Fold CV
        - **Comparison:** the **test** accuracy, precision **and** recall of one or more (at least two) **best classifiers** by plotting a bar graph. The user should specify the argument: accuracy, precision, recall **or** fl_macro, as the determining factor for plotting the best classifier(s) results.
        - Do these steps in a useful, intuitive way to get >50% mark for this task.

Clarity on what needs to be outputted: Draw the bar graph(s) the way you think is best for comparing accuracy, precision and recall for one or more specified classifiers. Show a (automated) final decision on the best combination of feature processing & classifier algorithms.

More Hints for more marks: You are not limited to what I lectured. Use your imagination and do research. *If you give yourself too low or high of a mark, I may penalise you!*

Deliverables:

- Python script that impresses me