=> Machine Learning Assignment 3 <=
150 MARKS

If you have any problems with this practical assignment, speak up well before the deadline!

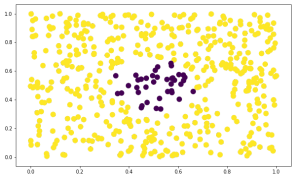
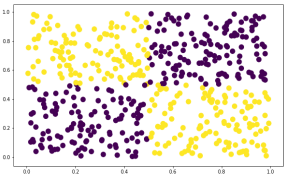
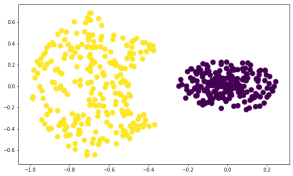
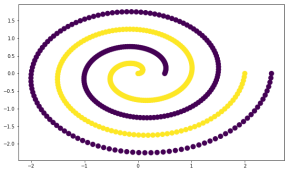
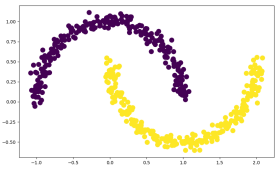
Deadline

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

Task 1: Which Kernel Works Best for SVM? [4 + 4 + 4 + 4 + 4 = 20 Marks]

The code provided in SVM_Exercises.py includes four different datasets and visualization code, which you should display as shown below. Complete the code by using the SVM kernel which will perform best in terms of accuracy on the data given in the code. Fix any errors, and you may 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 only aiming for the highest accuracy, and don't need to worry about your model generalising. Overfitting is allowed.).

Deliverables:

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

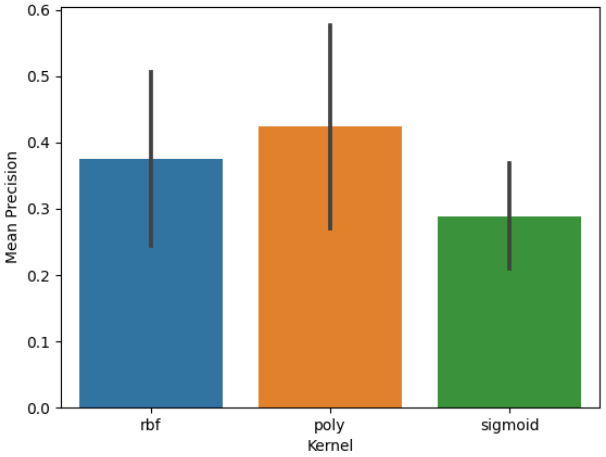
Task 2: Cats vs. Dogs with SVMs [5 + 10 + 5 + 10 = 30 marks]

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. Fit SVM models to allow for parameter estimation using grid search with 3-fold cross-validation. You are searching for the highest **precision score**. Use the following hyperparameters in your search:
    - C: 0.01, 1, 10
    - gamma: 1e-1, 1e-2, 1e-3
    - kernel: ‘rbf’, ‘poly’, ‘sigmoid’
3. The results should be formatted similar to this, 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'}`
4. Visualize the overall results **for each kernel** using a bar graph. Include the mean and standard deviation for each bar. Your bar graph layout should appear similar to this:



Deliverables:

- Python script with relevant print statements
- PNG of Bar Graph

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

The SVM model was not good enough! We need better (pre)processing for a better model.

Use your knowledge of everything you've learned up to now to try and outperform the SVM using MLP. Keep the SVM results from Task 2 in your new script, and just handle MLP afterwards.

- Use any combination of preprocessing you feel necessary to produce the highest precision.
- The more effort I see into your (pre)processing -> the higher the mark you get.
- *You must also report on the MLP results in the same format as Task 2*

Deliverables:

- Modified Python script, with prints to show results of all classifiers

Task 4: Preparing for the Final Boss [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)!

1. Take a look at multi_classifier.py. The user must specify two command-line arguments:
    - --dataset: Choices for *digits, kddcup99, iris, or wine* converted to **Pandas**
    - --classifiers: List classifier(s) as comma-separated arguments
2. Complete the script. The script should be able to do classification on the specified dataset using the specified classification algorithm(s) to get 50% of the marks:
        - k-Nearest Neighbours
        - Logistic Regression
        - Random Forest
        - XGBoost
        - Support Vector Machine

        - Multilayer Perceptron
3. Also 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 preprocessing such as scaling and selection/extraction, etc. argument
    - Pipeline to combine processes is recommended
    - Score the accuracy, precision **and** recall of a classifier using 5-Fold CV
    - Compare the **test** accuracy, precision **and** recall of up to two **best classifiers** by plotting a bar graph. The user specifies argument: accuracy, precision, recall **or** fl_micro, as the determining factor for plotting the best classifier(s) results.
    - Do these steps in a useful way to get >50% mark for this task.

TL;DR and Clarity on what needs to be outputted: Take in one or two classifiers as command-line arguments. If only one is specified, score the accuracy, precision and recall. If two classifiers are specified, draw the bar graph(s) the way you think is best for comparing accuracy, precision and recall. Show a final decision on the best combination of feature preprocessing & classifier algorithms.
More Hints for more marks: You may (and probably should) use some of the previously learnt methods for reading in any arbitrary dataset, and determining the best features + classifier combos, e.g. grid search or better. You are not limited to what I lectured. Use your imagination and do research. Make all command-line arguments intuitive to use or put an example use case for running via terminal. If you give yourself too low or high of a mark, I might penalise you!

Deliverables:

- Modified multi_classifier.py