Intelligent Systems Engineering
ECE 449
Fall 2023

Laboratory Exercise #1

Resources: *Machine Learning with Python*, from Cuantum Technologies (required text).

- Chapters 2.2-2.5 for the Pandas, Matplotlib, Seaborn and Scikit-learn libraries
- Chapters 3.1-3.5 for data preparation
- Chapter 4.2.4 for the *k*-NN algorithm

Wisconsim Breast Cancer Dataset, https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic

Box and Whisker Plots: https://asq.org/quality-resources/box-whisker-plot

Data Imputation: https://en.wikipedia.org/wiki/Imputation_(statistics)

Precision, Recall, F1 Score: https://en.wikipedia.org/wiki/Precision_and_recall

Summary: Using the scikit-learn libraries, build an AI pipeline that preprocesses the Wisconsin Breast Cancer dataset, trains a simple ML algorithm on it, and evaluates the performance of that algorithm.

Discussion: Modern AI systems in real-world usage can no longer be the product of a manual train-and-test process. The real world is constantly changing, while AI models ***don't*** change unless you force them to (by retraining them). New AI model versions must constantly be prepared, even while older versions are still running; when a new version outperforms the running one, it has to be swapped in right away. A recent empirical study at Google found that almost all of their hundreds of AI systems (only some of which are deep learning) are retrained at least once a day; a few systems may be retrained 100 times each day. Thus, data ingestion and preparation, parameter exploration, and model evaluation must now be highly automated. As we have discussed, these functions are grouped together in *AI pipelines*, and this is how AI is deployed in best-of-breed production systems at the most sophisticated companies.

Requirements: Using the scikit-learn libraries, construct a basic pipeline to ingest and prepare a small dataset, train and test a simple machine-learning algorithm on it, and evaluate the algorithm's performance. You may assume this pipeline is only supposed to produce one deployment model, and then terminate.

The dataset we will use is the Wisconsin Breast Cancer diagnostic dataset, available at the link above. This is a well-known small dataset for benchmarking shallow machine learning algorithms. It has two classes, benign and malignant, defined by a categorical variable (i.e. just names; you cannot order, add or multiply categorical values). There's an ID attribute that you need to throw away, and ten numerical attributes that you will use to predict the class of each record. You should begin processing this dataset by looking at the basic statistics of each attribute (median, 1st and 3rd quartile, min, max); this is best visualized as side-by-side box and whisker plots for each attribute. Matplotlib / Seaborn can be used to programmatically generate such a plot.

Certain values in these "predictor" attributes are missing, and must be filled in (i.e. data imputation). We will use mean imputation; specifically, replace each missing value with the mean value of that attribute *within the same class*. You will then need to rescale each attribute of your data to the interval [0,1] using the min-max scaling technique and the min & max values you found previously. Transform your class labels to 0 for benign, and 1 for malignant. Finally, you should randomly split your dataset into a training and testing set (preserve your class frequencies using stratified sampling). Don't forget to make sure your data is shaped properly for the learning algorithm!

We are going to train a *k*-Nearest Neighbors (*k*-NN) classifier to model this dataset. This is a simple algorithm with a single hyperparameter (*k*, the number of nearest neighbors to consider). The model predicts that the class of a test datapoint is the same as the class for the majority of the *k* datapoints closest to it in feature space. Your pipeline must automatically explore the performance of the *k*-NN algorithm for different values of k (for this exercise, $k \in (1..10)$ is enough). Compare the performance of the different *k*-NN versions on the test dataset using the F1 score, and report the performance of each version and your final conclusion on the best choice of the parameter *k*. Matplotlib / Seaborn can again be used to visualize your results; remember, in a corporate or research laboratory environment, you will need to persuade the actual decision-makers that your models were correct. Convince me! Presentation of your results throughout this project will count for 20% of your grade.

Submission: All of the above (box & whisker plot for all attributes; code to execute the data injection, preprocessing, and parameter exploration; presentation of the exploration results and final conclusion) should be automatically performed in a Python-based pipeline, and submitted as a Jupyter notebook. The TAs will grade your work by executing the notebooks on a fresh copy of the dataset, downloaded from the same source. Submit the notebooks via eClass in your lab section before the deadline.

Grading

Pipeline successfully executes all tasks:	80%
Presentation of results:	20%