# DATA.ML.200 Pattern Recognition and Machine Learning

*Exercise Set 2: October 28 – November 3, 2020*

Exercises consist of both pen&paper and computer assignments. Pen&paper questions are solved at home before exercises, while computer assignments are solved during exercise hours. The computer assignments are marked by text **`python`** and Pen&paper questions by text **`pen&paper`**

1. **`pen&paper`** *Design an optimal detector for step signal.*

   The lecture slides describe an optimal detector for a known waveform $s[n]$. Apply it to design the optimal detector for a step edge:

   $$s[n] = \begin{cases} -1, & \text{for } 0 \leq n < 5 \\ 1, & \text{for } 5 \leq n < 10 \end{cases}$$

   Simplify the expression as far as you can.

2. **`pen&paper`** *ROC and AUC.*

   A classifier is trained for cancer detection and the `predict_proba` method is applied on the test data. The probabilities of cancer for four patients (as assessed by the model) are shown in the below table. Draw the receiver operating characteristic curve. What is the Area Under Curve (ROC-AUC) score?

   | | **Prediction** | **True label** |
   |---|---|---|
   | *Patient 1* | 0.8 | 1 |
   | *Patient 2* | 0.5 | 1 |
   | *Patient 3* | 0.7 | 0 |
   | *Patient 4* | 0.1 | 0 |

3. **`pen&paper`** *Precision, recall and AP.*

   Draw the precision-recall curve for the above classifier and compute the Area Under Precision-Recall Curve (AUPRC) score.

4. **`python`** *Implement a sinusoid detector.*

   In this exercise we generate a noisy sinusoid with known frequency and see how the sinusoid detector of the lecture slides performs.

   a) Create a vector of zero and sinusoidal components that looks like the plot below. Commands: `np.zeros`, `np.concatenate`. Sinusoid is generated by `np.cos(2 * np.pi * 0.1 * n)`.

b) Create a noisy version of the signal by adding Gaussian noise with variance 0.5: `y_n = y + np.sqrt(0.5) * np.random.randn(y.size)`.

c) Implement the deterministic sinusoid detector (slide 20 of slideset 3)

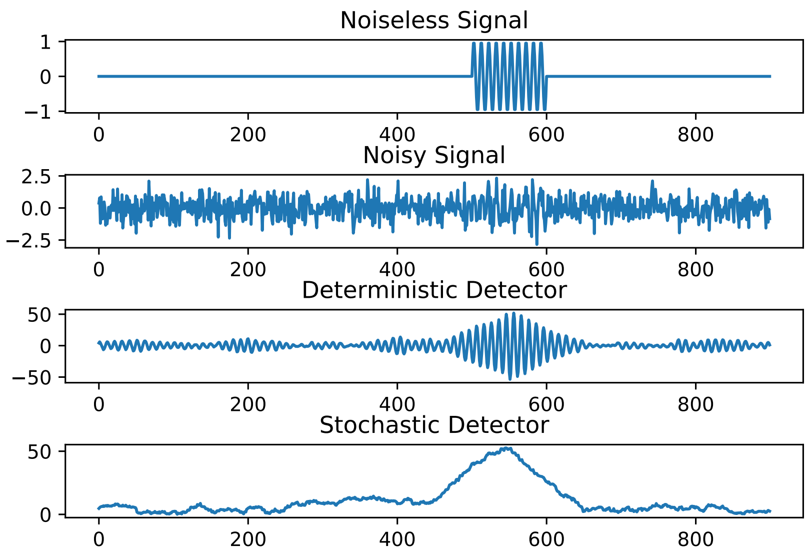
d) Implement the random signal version (slide 24 of slideset 3).

e) Generate plots like the ones below. Hint for plotting:

```
fig, ax = plt.subplots(4, 1)  # Create a figure with 4 axes

ax[0].plot(x)                 # This will be the topmost axis
ax[1].plot(x)                 # This will be the second axis
ax[2].plot(x)                 # This will be the 3rd axis
ax[3].plot(x)                 # This will be the 4th axis

plt.show()                    # Display on screen.
```



5. **python** *Train your first sklearn classifiers.*

In this exercise we will train a a number of classifiers and compare their performance. We will learn the details of the classifiers later, but for now just study the sklearn API (Google is your friend). Before you start, load the following dataset:

`http://www.cs.tut.fi/courses/SGN-41007/Ex1_data.zip`

a) Load the file `twoClassData.mat` to your Python workspace, and split the data into training and testing sets: samples `X[:200, ...]` for training and `X[200:, ...]` for testing.

b) Fill in the attached table (also find out how to make good looking tables in Jupyter).

| *Model* | *Accuracy Score* | *ROC-AUC Score* |
|---|---|---|
| *3-Nearest Neighbor* | | |
| *Linear Discriminant Analysis* | | |
| *Logistic Regression* | | |
| *Random Forest (`n_estimators=20`)* | | |

## Scikit-learn cheat sheet

- Create an instance of the classifier, *e.g.*,
  `model = KNearestClassifier(<params>)`
- Train the classifier: `model.fit(X_train, y_train)`
- Predict the classes for test data: `y_pred = model.predict(X_test)`
- (optional) predict class probabilities for test data:
  `y_pred = model.predict_proba(X_test)`
- Assess the model accuracy for the test data using the functions in `sklearn.metrics`