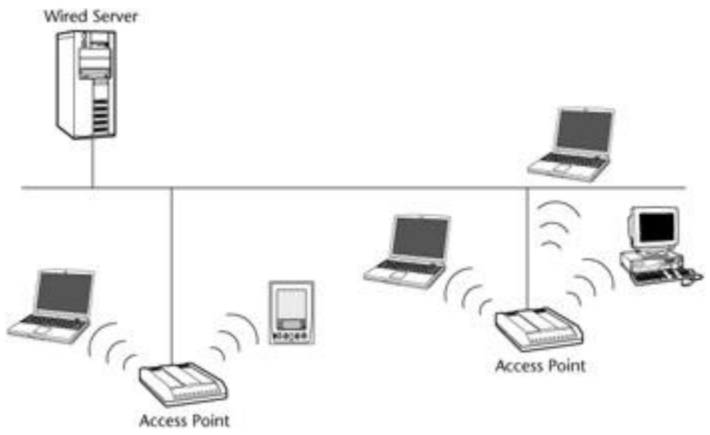
CodeLab – 3
Classification of WLANs power saving

Energy management is important for electronic devices and doubly so for portable systems that run on battery power. For this Code Lab, we consider the problem of controlling the power state of the wireless network interface controller. For optimal performance, this power state should adapt to the network usage. We will use a dataset published alongside a paper by Saeed and Kolberg, 2018 [1]. The researchers measured the traffic signatures associated with a number of applications on a mobile phone and labeled each according to the WLAN usage pattern. The dataset comprises 1350 samples with 6 features and a single classification label. The label (the type of usage) was assigned one of four categories: *high*, *varied*, *buffer*, or *low*.

In this Codelab, we will start with binary classification, where we will see the difference between weighted and non-weighted binary classification. Secondly, we will develop some multiclass models. There, we will see the difference between one-versus-all and multinomial classification. Various classification metrics will be used to assess the models' prediction capabilities.



Provide your answers in the Jupyter Notebook either as a comment or insert a markdown cell. Please print out your Jupyter Notebook in both PDF (or HTML if export to PDF is not possible) and ".ipynb" format then upload it to Brightspace. You will get points for each part separately and based on your overall performance, you will get the grades. Table 1 illustrates the points of each part and Table 2 shows the grading scheme of CodeLab 3.

Table 1: Points of CodeLab3

Part	Points
Task 1	2
Task 2	2
Task 3	3
Task 4	3
Bonus	1
Total	11

Table 2: Grading scheme of CodeLab3

Grade	Points
0	≤5
6	6-7
8	8-9
10	≥10

TASK 1 – Data Preparation (2 points)
In this lab, we will work on different classifiers for different classification models. Even though the data is already prepared, getting familiar with the data you will work with is important.

Please follow the following steps:

- Load "Network_Traffic.csv" into a Pandas data frame.
- Split the data frame into a feature data frame X and a label data frame y.

- Calculate and print the correlation matrix of X. (A correlation matrix is a square matrix where each element shows the correlation factor between the corresponding row and column variable.)
- Plot the heat map of correlation matrix X using the Seaborn library (*heatmap*).
- Plot the 2 (different) variables with the most correlation in a scatter plot
- Create an array named “y_{2c}” for two class classification. Convert the samples with the high label 1 and the rest 0.
- Calculate the ratio of high-class samples to total samples.
- Create an array named “y_{mc}” that can be used for multiclass classification. Use numerical labels according to the class, as follows: 3-High, 2-Varied, 1-Low, 0-Buffer.
- Plot the distribution of labels in the data set using a bar plot.
- Convert data frames X, y_{2c}, and y_{mc} to NumPy arrays (if they were Pandas objects). Convert the label vector elements to integers using the following notation, *y.astype(‘int’)*, where y is the label vector.
- Scale the feature matrix X with the *StandardScaler()* function from Sklearn.

Questions
1) Which variables have the highest correlation? Does this make sense to you? Why?
2) What is the ratio of high-class labels in binary classification?
3) What is the share of each label in multiclass classification?
4) Why might scaling improve the prediction capability of machine learning?

TASK 2: Binary classification with Logistic Regression (2 points)
In the second task, you will develop a binary classifier to detect whether the wireless network is operating in high consumption mode or not. ML models will be tested based on 25% of the input data. You can develop your own functions using Sklearn functions for tasks like training, predicting, or performance evaluation.

- Split the data (X and y_{2c}) into training (75%) and test sets (25%) using the “*train_test_split*” function from Sklearn. Use 4720 as the random state parameter to control the split.

The logistic regression model is a linear model to predict the probability of the sample belonging to a positive class ($y_{2c\_test} = 1$). The model utilizes the logistic function to perform the classification.

(Logistic regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

- Develop and train a logistic regression model with default parameters.
- Use the trained model to predict test labels.
- Calculate and print the following performance metrics: Accuracy, recall (sensitivity), precision, and F1 score.
- Plot the receiver operating characteristic (ROC) curve of training and test data in the same figure. By default, the logistic regression model uses a cut-off value of 0.5 to assign a positive or negative prediction to a data point. Different values can be used to bias the prediction towards negative or positive outcomes. The ROC curve plots the combinations of specificity and recall (sensitivity) that result from a range of cut-off values.
- Generate and plot the confusion matrix.

The class weight parameter adjusts the penalization factor of misclassified examples in model training. Depending on the application, the selection of weights varies. For example, in medical testing, the cost of false negatives (absence of a disease when it is present) outweighs the cost of false positives (incorrect indication of a disease when it is not present). In this binary classification, we need to modify the weights to reduce the number of missed high cases (false negatives).

- Develop and train a logistic regression model using the "*class_weight*" parameter to reduce the number of false negatives in the test data ($y_{2c\_test} = 1$, but $\hat{y} = 0$). (Tip: In CodeLab2, we constructed an SVM grid search. Similar to that, you don't have to try all the combinations yourself; you can build a nested loop and add a judgment statement at the end until you have a suitable result)
- Print out the weights you selected.
- Calculate and print the following performance metrics: Accuracy, recall, precision, and F1 score.
- Plot receiver operating characteristics (ROC) curve of training and test data in the same figure.
- Generate and plot the confusion matrix.

Questions
1) What differences do you find between train and test ROC curves in the logistic regression model?
2) What does the confusion matrix represent?
3) How do the weights influence the results of our classification in logistic regression?
4) What metrics are good to use to compare models? How can you decide what is a good model?

TASK 3: Binary classification with Support Vector Machines (3 points)
In this task, you will develop multiple support vector classifiers (SVC) for the binary classification task.

- Develop and train a linear SVC with the given parameters [C=1.0, coef0=0.0, tol=1e-3]. (Note: the default kernel is RBF and must be changed to linear with the option *kernel='linear'*)
- Calculate and print the following performance metrics: Accuracy, recall, precision, and F1 score.
- Plot the receiver operating characteristic (ROC) curves of training and test data in the same figure.
- Generate and plot the confusion matrix.

Next step, you will investigate the impact of hyperparameters. The regularization parameter (C) adjusts the penalty factor for samples in training.

- Develop two linear SVCs, one with parameter C as 0.0001 and one with C as 100, then train both models. (tol=1e-3)
- Calculate and print the following performance metrics for both cases: Accuracy, recall, precision, and F1 score.
- Plot the receiver operating characteristic (ROC) curves for both cases separately.
- Generate and plot the confusion matrices for both cases separately.

The other tunable hyperparameter is tolerance which adjusts the stopping criteria of the optimizer.

- Develop and train a linear SVC with the given parameters [C=1.0, coef0=0.0, tol=10].
- Calculate and print the following performance metrics: Accuracy, recall, precision, and F1 score.
- Plot the receiver operating characteristic (ROC) curves of training and test data in the same figure.

- Generate and plot the confusion matrix.

Kernels are decision functions applied to transform the feature space into higher dimensions. Nonlinear relationships between features can be found in high-dimensional space. In this CodeLab you will use a polynomial SVC (SVM classifier with a polynomial kernel).

- Develop and train two polynomial SVCs [*kernel='poly'*], one with the degree of 3 and one with the degree of 2. [C=1.0, coef0=0.0, tol=1e-3].
- Calculate and print the following performance metrics for both cases: Accuracy, recall, precision, and F1 score.
- Generate and plot the confusion matrices for both cases separately.

Questions
1) Compare linear SVM and logistic regression models.
2) What did you observe when parameter C is varied? Explain the role of parameter C in SVC.
3) What did you observe when tolerance is changed to 10? Explain the role of parameter tolerance in SVC.
4) Which SVM model do you prefer for this problem, polynomial or linear kernel? Why? What is the potential benefit of using a nonlinear kernel?
5) Rank the models from the highest performance to the lowest one according to their F1 scores.

TASK 4: Multi-class classification (3 points)
Multi-class classification predicts the best among a set of labels (four, in this case). We consider two different model classes (logistic regression and SVM) and within the logistic regression model, both the One vs. All and the multi-class approach.

The One vs. All (called 'one versus rest' in sklearn) method constructs binary classifiers for each class. The final prediction is based on the model with the highest score or the largest (signed) distance from the classification boundary.

- Split the data (X, and y_{mc}) into training (75%) and test sets (25%) using the "*train_test_split*" function from Sklearn. Use 4720 as the random state parameter to control the split.
- Develop and train a logistic regression model using the one vs. rest (parameter *multi_class='ovr'*) method using the "y_{mc}" label vector.
- Calculate and print the accuracy of the model.
- Generate and plot the confusion matrix.

For logistic regression, we can also directly train a multi-class classifier using the multi-class cross-entropy loss (sklearn calls this the multinomial cross-entropy loss).

- Develop and train a multi-class logistic regression model (parameter *multi_class='multinomial'*)
- Calculate and print the accuracy of the model.
- Generate and plot the confusion matrix.

SVM can be used in multi-class classification tasks as well, but it will rely on the one-versus-all approach (or, for non-linear kernels, on the related one-versus-one approach).

- Develop and train a linear SVC [kernel='linear'] as a one vs. all classifier with the given parameters [C=1.0, coef0=0.0, tol=1e-3].
- Calculate and print the accuracy of the model.

- Generate and plot the confusion matrix.

Questions

1) How does the performance change for both classification methods for the logistic regression and why?
2) Which classification method is more computationally demanding and why?
3) Why can other classification metrics not be used directly in multi-class classification tasks?
4) Why can multi-class classification be directly applied to logistic regression but not SVM?
5) Compare all three models' performance in multi-class classification.
6) Which classes have higher mismatches in logistic regression?
7) Which classes have higher mismatches in SVM?

Bonus Task (1 point)

Conduct a multi-class classification using only binary classifiers. You can use either logistic regression or SVM. Describe the steps you follow and how it works. Calculate each classifier's output probabilities for the test data and plot them in ascending order.

References

[1] Saeed, Ahmed, and Mario Kolberg. "Towards optimizing WLANs power saving: Novel context-aware network traffic classification based on a machine learning approach." *IEEE Access* 7 (2018): 3122-3135.