INF8245E(Fall 2024) : Machine Learning — Given on: Oct 23^{th}

Assignment #3 — Due on : Nov 15, 10 pm

- This is an individual assignment. You are not allowed to discuss the problems with other students.
- Make sure to write your code only in the .py files provided. Avoid creating new files. Do not rename the files or functions as it will interfere with the autograder's ability to evaluate your work correctly. Also, do not change the input or output structure of the functions.
- When Submitting to GradeScope, be sure to submit
  1. A '.zip' file containing all your python codes (in .py format) to the 'Assignment 3 - Code' section on Gradescope.
  2. A 'pdf' file that contains your answers to the questions to the 'Assignment 3 - Report' entry.
- Part of this assignment will be auto-graded by Gradescope. You can use it as immediate feedback to improve your answers. You can resubmit as many times as you want. We provide some tests that you can use to check that your code will be executed properly on Gradescope. These are **not** meant to check the correctness of your answers. We encourage you to write your own tests for this.
- Please provide the title and the labels of the axes for all the figures your are plotting.
- If you have questions regarding the assignment, you can ask for clarifications in Piazza. You should use the corresponding tag for this assignment.
- Before starting the assignment, make sure that you have downloaded all the data and tests related for the assignment and put them in the appropriate locations.
- For this assignment, you are allowed to use libraries with general utilities, such as numpy and pandas, and you are expected to use pre-existing implementation of the algorithms available in scikit-learn.
- You cannot use ChatGPT or any other code assistance tool for the programming part, however if you use ChatGPT to edit grammar in your report, you have to explicitly state it in the report.
- For the parts marked with **Report** make sure that you have answered them in your report.
- For the coding part, you should implement your codes in the given python files. Do not change their names or function names. The auto-grader in Gradescope takes time around 5 minutes. Give it time until it will be finished.

Data Exploration and Preprocessing (15 Points)

1. In this assignment, you will employ several supervised learning algorithms to accurately model whether a client will subscribe to a term deposit using data from a direct marketing campaign of a Portuguese banking institution. You will then choose the best candidate algorithm from preliminary results and further optimize this algorithm to best model the data. Your goal with this implementation is to construct a model that accurately predicts whether a client will subscribe to a term deposit based on various features such as age, job type, and contact duration. This task is crucial in a business context where understanding client behavior can help optimize marketing strategies and improve customer engagement. The dataset for this project originated from the UCI Machine Learning Repository and was donated by Moro et al. [1]. The dataset has been loaded by running the function **downloading_data** in **q1.py** python file. You may want to print a few rows of the dataset to see the type of features and label.

   In **q1.py**, data is loaded in **bank_marketing**, **X** shows the feature matrix and **y** shows the labels.

   First, we need to perform an initial exploration of the dataset to understand its structure.

   (a) (6 Points) Show:

   - number of samples
   - number of clients subscribed to a term deposit,
   - percentage of clients subscribed to a term deposit.

   Write your code in *data_exploration* function in **q1.py**.

   (b) (2 Points) Separate features and labels in the data. Write your code in *extract_features* function in **q1.py**

   (c) (5 Points) Convert categorical variables to numerical variables: Some features might not be numerical called categorical variables. However, the learning algorithms we want to use work with numerical variables. One way to convert categorical to numerical variables is to use *LabelEncoder* of *sklearn*. Write your code in *feature_encoding* function to convert variables in **q1.py**.

   Hint: The non-numerical features' names are stored in *non_numerical_columns_names* in the *feature_encoding* function.

   (d) (2 Points) Moreover, the labels are also categorical since they are either "yes" or "no". So convert "yes" to "1" and "no" to "0". Write you code in *encode_label* function to convert labels in **q1.py**.

[1] A data-driven approach to predict the success of bank telemarketing

Model Training and Evaluation (55 Points)

In this part, we want to make train and test sets and then pick some classification algorithms to train on the training set.

Now we need to have train set to train the model on and test set which is separate from the training set to evaluate the model. For this we split the data (both features and their labels) into training and test sets.

1. (5 Points) Shuffle and Split Data: Use an 80-20 split ratio meaning 80% for training and 20% for test. For this, it is important to shuffle data before splitting that or use a function that does it. Also to have the unique results set random seed to 0. To do that implement your code in function *data_splits* in **q2.py**. You need to set **random state of 0**.

   Hint: Use *train_test_split* from *sklearn*. You need to set random state of 0 in *train_test_split*.

2. (5 Points) Normalize features: normalization ensures that each feature is treated equally when applying supervised learners. Write you code in *normalize_features* function to normalize the variables in **q1.py** . You should calculate the normalization parameters only using the training data and then use the same parameters to normalize the test data.

   Hint: Use *MinMaxScaler* from *sklearn*.

3. (10 Points) Train models: now we need to train a model on the training set to learn how to predict the labels from the features. We chose these three classification algorithms to model the data:

   - (2 Points) Decision-Trees,
   - (2 Points) Random-Forest,
   - (2 Points) Support Vector Machines (SVM with Support Vector Classification).

   For each of these methods you need to train models on all the samples of the training set. You need to set random state of 0 for each model. You should use the Scikit-Learn package. Implement your code in function *train_model* in **q2.py**

   Note 1: Use the default settings for each model — you will tune the models in a later section.

   Note 2: Depending on the algorithms, the code may take some time to run!

4. (35 Points) Model evaluation:

   Report: You need to write the results of these questions in your report in addition to the implementation.

   Here we want to use metrics to see first if the models with default parameters learned the data and then evaluate the performance of the methods.

(a) (10 Points) Calculate accuracy and F1-score to evaluate the performance of each model with default values. Write your code in function *eval_model* in **q2.py**. First calculate the accuracy score for both the training set and test set. Then, calculate the F-score for both sets.

(b) (10 points) You need to report these values for both train and test sets in your report. Justify your response!

(c) (5 points) Use classification_report() and confusion_matrix() from Scikit-Learn. Implement your code in *report_model* function in **q2.py**.

(d) (10 points) Write your results from classification_report(), confusion_matrix() in your report. Explain what they mean!

Model Tuning (45 Points)

Here you will fine tune parameters of the three supervised learning models to find a better model for each of them. You will then perform a grid search optimization with cross validation for the model over the entire training set (X_train and y_train).

1. (20 Points) Specify Hyper-Parameters:

   Report: You need to write the answer of this question in your report.

   For Random-Forest and SVM, write five significant hyper-parameters they have. Then for each of the hyper-parameters explain what they are and what are their impact on their model briefly.

2. (20 Points) Grid search: use grid search *(GridSearchCV)* for the hyper-parameters of the models with a good range of values. You are responsible for specifying a reasonable range for each hyper-parameter and find at least one setting which has higher performance than the default.

   (a) (3 points) Create a dictionary of the specified parameters below for each models with a good range of values. Here are the parameters for each model:

      1. Decision-Trees:*criterion, max_depth, min_sample_leaf, max_leaf_nodes*
      2. Random-Forest: *n_estimators, max_depth, bootstrap*
      3. SVM: *kernel, shrinking, C(Regularization parameter), tolerance, gamma*

   (b) (3 points) Initialize the classifier with 0 random state.

   (c) (2 points) Create a scorer, it should be accuracy.

   (d) (2 points) In function *perform_grid_search* write your code for cross validation with 10 fold. You need to use StratifiedKFold.

   (e) (2 points) Perform grid search on the classifier using the *scorer* and *cross-validation*, and store it in *grid_search*.

   (f) (6 points) Fit the grid search object to the training data (X_train, y_train).

(g) (2 points) Print the best parameters and best scores.

Implement your codes in **q3.py**.

3. (5 Points) **Report:** explain why do we need cross validation and why should we do stratified cross-validation in your report?

Report (10 Points)

You need to write the answers of these questions in your report based on the previous question.

1. (6 Points) For each of the models, for the below hyper-parameters and values, calculate the accuracy on the training set. Then make a plot for each of the models where x-axis is the value of the hyper-parameter and y-axis is the accuracy. Justify the effect of the hyper-parameter values on the accuracy for each model separately.

   - Decision-Trees, for max_depth ={None, 5, 10}
   - Random-Forest, for n_estimators ={5, 10, 30}
   - SVM, for kernel ={linear, poly, rbf}

2. (2 Points) With the best hyper-parameters values you found from the model tuning part, now find the accuracy of test dataset. You need to find it for each model separately. Then plot it such that x-axis is model names and y-axis is the test accuracy.

3. (2 Points) What is the best model's accuracy on test across all models? Why do you think it is better? (explain it in one paragraph)