Supervised : Classification

1 Classification

1. Classification: getting started
   (a) Load the clean dataset of used cars
   (b) Let's say that we would now like to predict if a car costs less or more than 12.000. Replace the column `price` by a column `12k+`, which is 1 if the value is above 12.000.
   (c) Train a Logistic Regression and a Decision Tree Classifier(limit leafs=20).
   (d) Compare the results (on a test set) using the classification scores we have seen in class: Precision, Recall, Accuracy, F1-score.
   (e) Compute also the Average Precision and the AUC. Since they are based on a ranking of scores, you need to use the `predict_proba` function of classifiers to compute scores.
   (f) Plot the confusion matrix (You can compute it manually or use `confusion_matrix` method of sklearn.) You can use the method `heatmap` of seaborn. Plot the raw values or normalized confusion matrix (check `confusion_matrix` documentation). Try to interpret.
   (g) Use KNN too (it is in sklearn).
   (h) This dataset is not balanced: compare the proportion of elements of both classes.
   (i) Create a balanced test set by subsampling (i.e., keep as many elements from the most numerous class than from the less numerous one). Recompute some results on this balanced dataset and check the change in scores.
2. Regularization
   (a) Train first a classic linear regression, to predict the price. Plot its performance (at least MAE and R2) and its coefficients.
   (b) Train a `Lasso` regressor, first with the default `alpha` parameter of 1. Plot its performance, coefficients, and compare with non regularized result.
   (c) Do the same with a `Ridge` regressor.
   (d) To see the effect of parameters on coefficients, make the parameter vary from 1 to 10 in log scale, and plot the evoluation of coefficients. You can take inspiration from `https://scikit-learn.org/stable/auto_examples/linear_model/plot_ridge_path.html`
   (e) Do the same with Lasso, with the appropriate range of parameter.
3. XGboost
   (a) As a baseline, train a Decision Tree, with a quick manual calibration of parameters to get a decent result
   (b) XGBoost is provided in an independent library called `xgboost`. However, it provides a sklearn compatible interface. Check `https://xgboost.readthedocs.io/en/stable/python/python_api.html#module-xgboost.sklearn` for its reference
   (c) Train XGBoost with default parameters on your data. Compare results with other methods.

2 Advanced

4. Parameter tuning

(a) For your favorite method, try automatic parameter tuning, using the GridSearch tool of sklearn `model_selection.GridSearchCV`.

(b) Start by exploring on a single parameter, plot the curve of the different scores according to the parameter value.

(c) Using a method having at least 2 parameters, plot a heatmap (or a contour chart, i.e., 2D kdeplot) to visualize the parameter space performance

5. Real data

   (a) Advanced methods such as XGboost or lasso makes a real difference on complex datasets, not so much on toy ones like these ones. Try with real datasets proposed in the class page.