Supervised : basics

The objective of those exercises is to practice the basics of supervised machine learning.

1 Fundamentals

1. Preparing the dataset
   (a) Get ready to use the same dataset as for the previous class. Keep columns [budget, popularity, revenue, runtime, vote_average, vote_count] as numerical columns. Convert 0 in budget, revenue, runtime and vote_count into na.
   (b) An important question to address is that of missing values. Using `missingno` package, visualize your missing values, in the original dataframe and in the numerical values only to make informed decisions. Check the documentation `https://github.com/ResidentMario/missingno` to see what you can do.
   (c) Check visually if movies with more votes tend to have less missing values, and if more recent movies have less missing values (you can sort a dataframe using `.sort_values`.
2. Classification: getting started
   (a) Let's say that we would now like to predict if a movie will be profitable or not. Compute a new column defined as revenue - budget, and transform into a boolean value, True if positive, False if negative. Be careful with missing values...
   (b) Train a Logistic Regression and a Decision Tree Classifier(limit leafs=50).
   (c) Compare the results using the classification scores we have seen in class: Precision, Recall, Accuracy, F1-score.
   (d) 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.
   (e) 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.
   (f) Use KNN and Naive Bayes methods too.
   (g) This dataset is not balanced: compare the proportion of elements of both classes.

2 Advanced

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

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

(d) 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

3 Going Further

(a) Do you think we could improve on those scores? Discuss with your peers, and propose solutions based on feature engineering. You can use other columns, or transform some columns, play with parameters...