Machine Learning
Basic Practice

Feature Engineering: **Transform raw data into the dataset**

Categorical data => numerical:

$$\begin{aligned} \text{red} &= [1,0,0] \\ \text{yellow} &= [0,1,0] \\ \text{green} &= [0,0,1] \end{aligned}$$

- Cannot simply use 1, 2, 3 because it implies an order.

Binning: Transform numerical data into categorical ones

Normalization:

$$\bar{x}^{(j)} = \frac{x^{(j)} - min^{(j)}}{max^{(j)} - min^{(j)}},$$

- **Standardization/z-score normalization**:

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}}.$$

    - Normalize the distribution to have μ = 0 and σ = 1
- Choose standardization:
    - Unsupervised learning
    - Bell shaped feature
    - When have outliers
    - Otherwise normalization

Data Imputation - Deal with missing feature

1. Replace the missing value of the feature with the average value of this feature in the dataset.
2. Set the missing value to be a value outside of the value range of the feature, and let the computer to learn how to deal with outliers.
3. Use missing values as targets

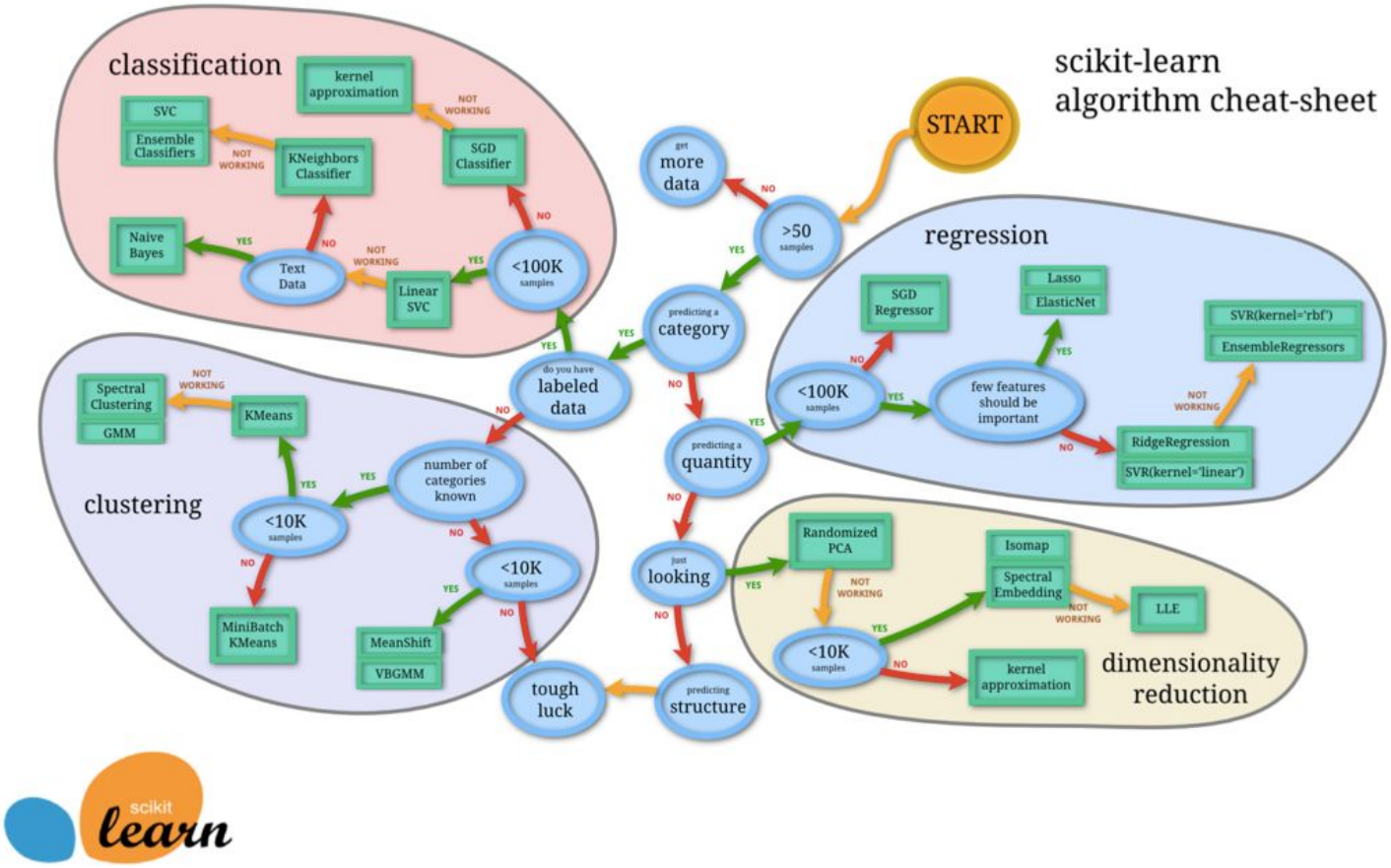
Learning Algorithm Selection

1. In-memory vs. out-of-memory
    - Can the data be fully loaded into the RAM?
    - If not choose incremental learning algorithms
2. Categorical vs. numerical
3. Linear separable?

- Yes - SVM w/ linear kernel, logistic regression
- No - NNWs etc.

4. Prediction speed:
    - KNN is slow, so does deep/recurrent networks.

Scikit-Learn Algorithm



Regularization: Deal with overfitting

$$\min_{\mathbf{w},b}\left[C|\mathbf{w}| + \frac{1}{N}\sum_{i=1}^{N}(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2\right],$$

- C ::= Hyperparameter
    - “Determines the tradeoff btw increasing the margin and ensuring each x lies on the right side.”
- Usually chosen experimentally

- As C gets greater, the cost for misclassification (the second term of the function) becomes negligible. => SVM will try to find the highest margin by ignoring misclassification.

Model Performance Assessment

1. For REGRESSION:
    - **Mean Squared Error**
2. For CLASSIFICATION:
    a. **Accuracy / Cost-sensitive accuracy:**

$$\text{accuracy} \stackrel{\text{def}}{=} \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

    b. **Precision:**

$$\text{precision} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

        - "The proportion of relevant documents in all returned documents".
        - Useful when solving a spam problem. People have more tolerance of not spamming junk mails over spamming important emails.

    c. **Recall:**

$$\text{recall} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

        - "The ratio of the relevant documents returned o the total number of the relevant documents that could have been returned."

    d. **Confusion Matrix:**
        - Useful when accessing multi-class classification.
        - In its (i,j) cell, it shows the number of instances i that were predicted to be in class j.

e. Area under the ROC Curve (AUC) ~

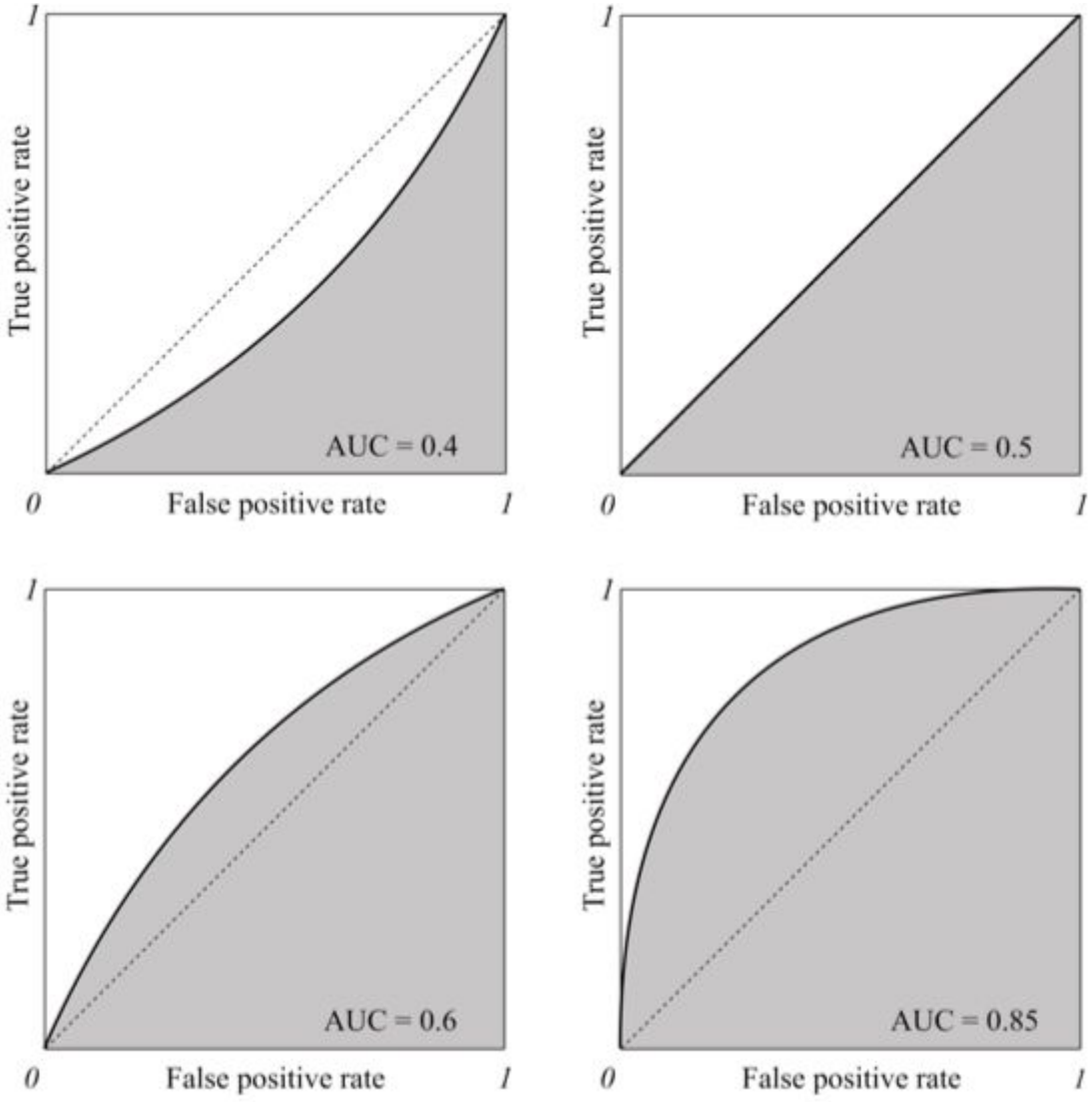


Figure 3: The area under the ROC curve (shown on gray).

- 1. Discretize the range of the confidence score.
    - If this range for a model is [0, 1], then you can discretize it like this: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1].

  2. Use each discrete value as the prediction threshold, predict the labels of examples in your dataset using the model and this threshold.

- Can only be used to assess classifiers w/ return confidence score/probability of prediction
    - E.g. Logistic Regression, NNWs, Decision Tree
- The higher the AUC the better the classifier

Hyperparameter Tuning

How to select good hyperparameters like C for SVM?

1. **Grid Search**
    - Use different different non-random value for different models and evaluate them to find the best one.
2. **Random search** and **Bayesian hyperparameter optimization.**
3. **Cross-Validation**
    - Build model iteratively by dividing the dataset into subsets (called folds) and average them to get the final result.