CS-EJ3211 Machine Learning with Python

Session 4 – Classification

Alexander Pavlyuk

Aalto University
FITech

21.06.23

Categorical vs. Numeric labels

Numeric labels:

- Regression problem (week 2).
- Structured label space
- Example: the real numbers $\mathbb{R}$.

Categorical labels:

- Classification problem.
- Finite label space consists of classes/categories.
- Example: phone storage condition is "Empty", "Partly filled", or "Full".

Ordinal labels:

- Classification or regression problem.
- Finite and structured label space.
- Example: $y \in \{1, 2, 3\}$.

Categorical labels

Binary classification – each data point belongs to exactly one out of two different classes.

a cat

not a cat

Multiclass classification – each data point belongs to exactly one out of more than two different classes.

a lemur

a parrot

a Komodo dragon

Classification performance

Possible outcomes of binary classification:

$$y \in \{0, 1\}, \text{ where}$$
$$y = 1 \text{ is positive class}$$
$$y = 0 \text{ is negative class}$$

- $y = 0,\ \hat{y} = 0$ True Negative (TN)
- $y = 0,\ \hat{y} = 1$ False Positive (FP)
- $y = 1,\ \hat{y} = 0$ False Negative (FN)
- $y = 1,\ \hat{y} = 1$ True Positive (TP)

Classification performance metrics

Accuracy – fraction of correctly predicted labels.

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Precision – fraction of correctly predicted positive class among all predicted positive.

$$\frac{TP}{TP + FP}$$

Recall (sensitivity) – fraction of correctly predicted positive class among all with true label positive.

$$\frac{TP}{TP + FN}$$

F1 score – combination of precision and recall. High F1 score implies low FP and low FN.

$$2 * \frac{precision * recall}{precision + recall}$$

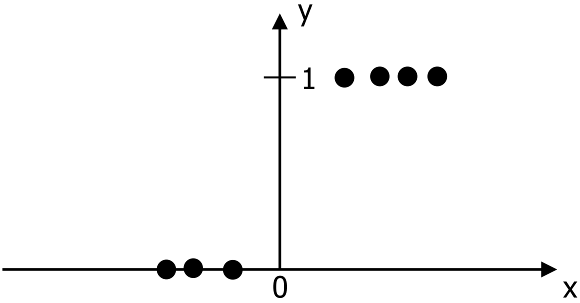
Classification methods – Logistic regression

Logistic regression is a binary classification method that learns a hypothesis out of the linear hypothesis space.

$$\mathcal{H}^{(n)} := \{h^{(\boldsymbol{w})}: \mathbb{R}^n \longrightarrow \mathbb{R}: h^{(\boldsymbol{w})}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} \text{ with some vector parameter } \boldsymbol{w} \in \mathbb{R}^n\}.$$

Nominal classes can be encoded in binary:

Positive diagnosis, negative diagnosis → $y \in \{0, 1\}$

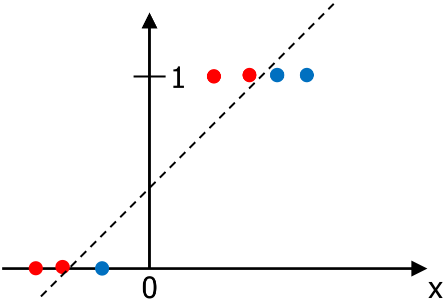


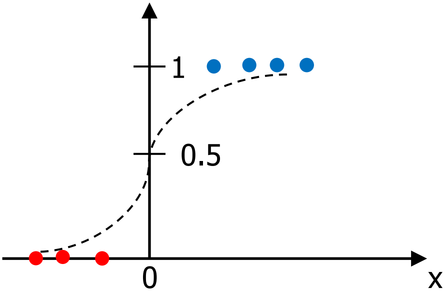
Linear vs. Logistic regression

Linear regression:
$$h(x) = \vec{w}^T x$$

Logistic regression:
$$h(x) = \frac{1}{1 + e^{-\vec{w}^T x}}$$

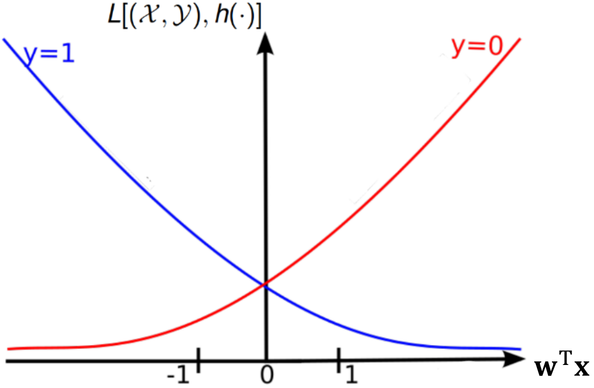
$$y = \begin{cases} 1 & \text{if } h(x) \geq 0.5 \\ 0 & \text{else} \end{cases}$$





Logistic loss

$$L[(\mathcal{X},\mathcal{Y}),h(\cdot)] = \begin{cases} -\log(h(x)) & \text{if } y=1 \\ -\log(1-h(x)) & \text{else} \end{cases}$$



Data standardization

Definition: the process of rescaling the data so that the mean is zero and the variance is one.

Process (for feature matrix **X**): for all elements in each column, we subtract the column mean (μ) and divide by the standard deviation (σ) of the column.

$$\boldsymbol{X} = \begin{matrix} x_1^1 & \cdots & x_k^1 \\ \cdots & \ddots & \cdots \\ x_1^n & \cdots & x_k^n \end{matrix}, \text{where}$$

$$\mathbf{z}_\mathrm{j}^{(\mathrm{i})} = \frac{\boldsymbol{x}_j^{(i)} - \mu(\boldsymbol{x}_j)}{\sigma(\boldsymbol{x}_j)}$$

n is the length of each feature vector,
k is the number of features.

Student Task 4.1 – Logistic Regression

Create Standard_Scaler object.

 # scaler = ...

Create LogisticRegression object.

 # log_reg = ...

Create Pipeline object.

 # pipe = ...

Fit the Pipeline to the training set.

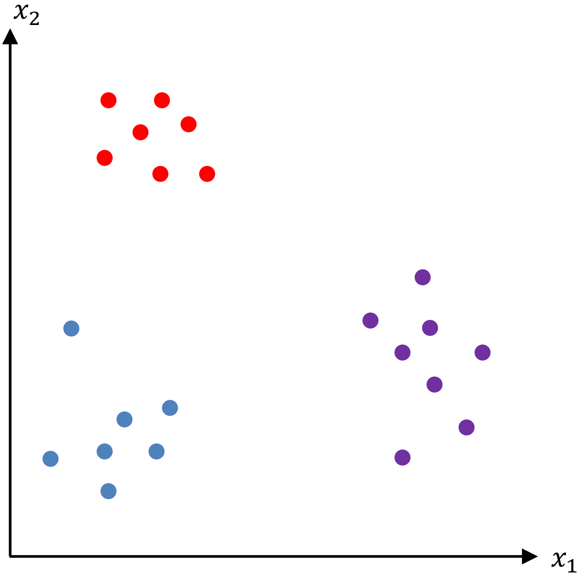
 # pipe.xxx(...)

Compute training and testing accuracies by calling .score(...) method.

 # acc_train = ...

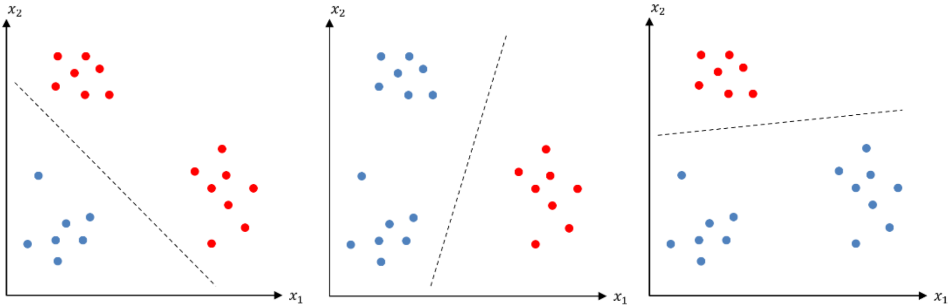
 # acc_test =...

Multiclass Classification

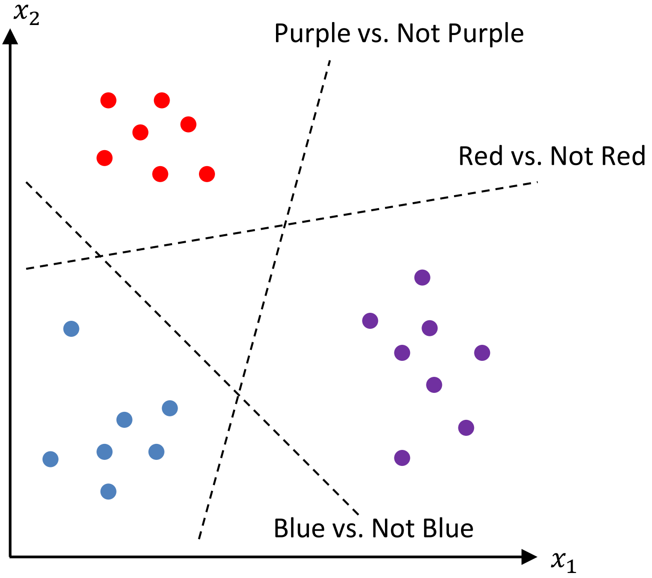


Multiclass Classification

Divide the multiclass classification problem into several binary classification subproblems



One-vs-Rest (OVR) strategy



Student Task 4.2 – Tuning a Logistic Regression model

Create the Pipeline object (remember to specify multi_class parameter as “ovr” in LogisticRegression inside the Pipeline).

 # pipe = …

Create a parameter dictionary containing one key-value pair of the parameter “C”.

 # params = …

Create GridSearchCV object.

 # cv = …

Perform 5-fold cross-validation. Remember to use training dataset!

 # cv.fit(…)

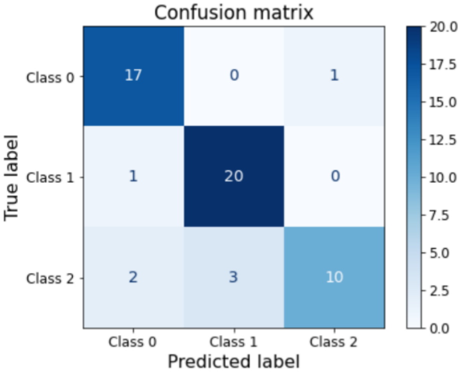
Store the average training and validation accuracies. GridSearchCV.cv_results_ attribute contains a dictionary with the performance data. Extract the required data by the proper key name.

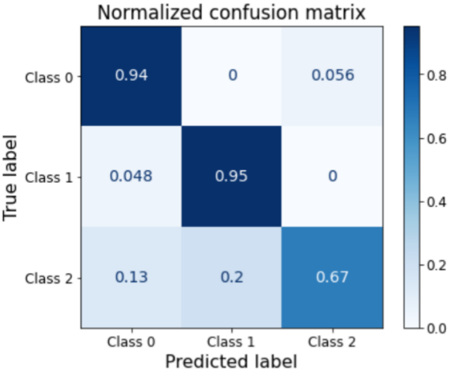
 # acc_train = …
 # acc_val = …

Store the best estimator by calling GridSearchCV.best_estimator_ attribute.

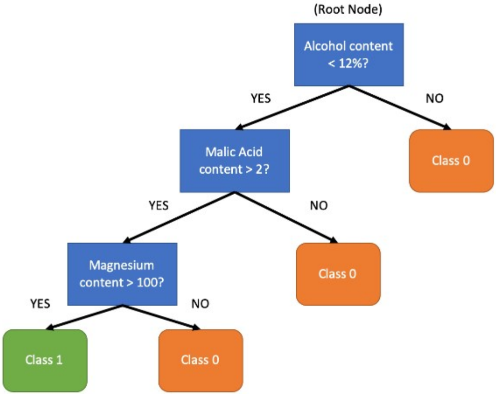
 # best model = …

Confusion Matrix





Decision Tree



Decision Trees consist of

- Decision (or test) nodes.
- Branches.
- Leaf nodes.

See MLBasics book section 3.10 for details

Student Task 4.3 – Decision Tree Classifier

Create Decision Tree Classifier object.

 # clf = …

Fit the Decision Tree Classifier to the training set.

 # clf. …

Compute training and testing accuracies by calling .score(…) method.

 # acc_train = …

 # acc_test = …

Decision Tree - Regularization

Hyperparameters available for tuning:

- Maximum depth of the tree.
- Minimum number of data points in leaf nodes.
- The minimum number of samples required to split an internal node.
- The number of features to consider when looking for the best split.
- Maximum number of leaf nodes.

See sklearn docs for more options and detailed explanations (link).

Random forest:

- Ensemble model with multiple decision trees.
- A data point is classified using a consensus based on the predictions of all decision trees in the “forest”.

Logistic Regression vs. Decision Tree

Logistic regression:

Pros:
- Minimizing a logistic loss amounts to a smooth convex optimization problem (gradient-based method are possible for application).
- Good for linear relationships between the predictors and response.
- Good for the small sample size.

Cons:
- Data pre-processing is required.
- Poor performance on complex data with outliers, non-linear relationships and other.

Decision Tree:

Pros:
- Very interpretable.
- No need for data pre-processing.
- Flexible hypothesis space, including complex predictors.

Cons:
- Prone to severe overfitting.
- Training is not as efficient, and globally optimum solution is not guaranteed.
- Not robust to changes in data. Small changes can result in very different models.

The End

Questions?