

Classification task: other models

There are several models that perform Classification, each with a different behavior.

We briefly explore some of the differences.

Decision boundaries

Just as we saw for the Regression task:

- there are multiple models for solving a Classification task

For Binary Classification

- the models create different decision boundaries

Output: probabilities or just classes ?

The ultimate output of a classifier

- is a single class label
- prediction of the class to which the example belongs

But many Classifiers output a *probability distribution* over all the classes

$$p(\mathbf{y} \mid \mathbf{x})$$

where $p(\mathbf{y} \mid \mathbf{x})$

- is a vector whose length is the number of classes

There are several possibilities for converting the probability vector to a single class

- choose the class with highest probability
 - e.g., in KNN
 - we chose the class for a test example
 - by comparing against k training examples
 - and choosing the class c that whose label was most frequent among the k examples
- for Binary Classification
 - compare probability of the Positive class to a threshold
 - choose class "Positive" only if the predicted probability of Positive exceeds the threshold

Some (but not all) classifiers in `sklearn`

- implement a method `predict_proba`
 - that returns the probability vector

For Classifiers that return probability vectors

- the ultimate class label predicted
- *can be adjusted by the user*

Here is some pseudo-code:


```
# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Get predicted probabilities for new data
probabilities = model.predict_proba(X_test)[:, 1] # Probability of positive class

# Set a custom threshold (e.g., 0.7)
custom_threshold = 0.7

# Make predictions based on the custom threshold
predictions = (probabilities >= custom_threshold).astype(int)
```

In the [Precision/Recall tradeoff \(Error Analysis.ipynb#Precision/Recall-Tradeoff\)](#) module

- we examined the effect of changing the threshold
- on conditional Performance Metrics
 - recall, precision

There is a [good discussion \(https://scikit-learn.org/stable/modules/classification_threshold.html\)](https://scikit-learn.org/stable/modules/classification_threshold.html) on adjusting the probability threshold in the sklearn documentation.

Reliability diagrams

We can also compare Classifiers

- by comparing the predictions
- across a wide range of examples

For Classifiers that produce probability distributions

- how confident is the Classifier in its prediction ?
 - examine the distribution of probability mass across many examples

A confident Classifier's distribution is bimodal

- most of the probability mass near 0 or 1

Another property:

How reliable is the prediction ?

- for Binary Classification
- examine all the examples assigned predicted probability $\hat{p} = P$ of being Positive
- the fraction p of these examples whose true labels are Positive
- should be close to \hat{p}

A plot of p versus \hat{p} is called a *Reliability diagram*

See [here \(https://scikit-learn.org/stable/modules/calibration.html\)](https://scikit-learn.org/stable/modules/calibration.html) for explanation and [here \(https://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html\)](https://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html) for code.

In [8]: `print("Done")`

Done

