1 Module 12: Classification and KNN

Contents

Regression: Given a set of features, predict a real-valued outcome
ClassificationL Given a set of features, predict the class of a sample

1.1 K Nearest Neighbors Classifier

Essentially: "Which data point in the training set is *nearest* to the sample?"

Number of neighbors: Calibrates the number of training set points which decide the nearest sample.

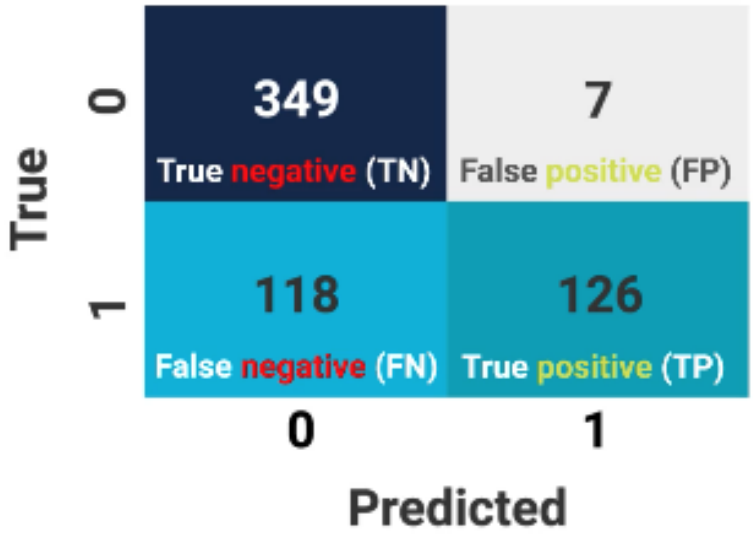
Decision boundary: the boundary surface between clusters – the complexity of the boundary suggests the degree of overfitting. Increasing neighbors simplifies the decision surface.

$$\text{Complexity} \propto \frac{1}{k} \text{for } k \in [1, n]$$

```
from sklearn.neighbors import KNeighborsClassifier

model - KNeighborsClassifier(n_neighbors = int)
model.fit(X,y)
model.predict(X)

from sklearn.metrics import accuracy_score

# Accuracy
accuracy_score(model.predict(X), y)
# Misclassification rate
1 - accuracy_score(model.predict(X), y)

# Get model level of confidence in prediction
model.predict_proba(X,y)
# Returns array with first column corresponding to confidence in class 0;
↪  second in class 1
[[0.7, 0.3],
 [0.9, 0.1], etc.]
```

Accuracy is not always the right metric – if the number of samples in each class is imbalanced, the accuracy may be misleading

Confusion matrix



- Accuracy: (TP + TN)/(TN + FP + FN + TP)
- Precision: TP/(TP + FP)
  How many who tested were predicted positive, are actually positive?
- Recall (or Sensitivity): TP/(TP + FN)
  Of those who are positive, how many were correctly predicted positive?
- Specificity: TN/(TN + FP)
  Of those who were negative, how many were correctly predicted negative?
- F1 (weighted avg. of precision & recall): $\frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$
  Used for uneven class distributions.

Precision-recall curve:

```
from sklearn.metrics import precision_recall_curve
```

ROC curve (Reciever operator characteristic): False Positive Rate (1 - Specificity) vs. True Positive Rate (Recall)
Integrable as a score i.e. for GridSearchCV(scoring='roc_auc')

1.2 K Nearest Neighbors Regressor

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors = int)
```

k=1 is a step function following every single data point; k ¿¿ 1 appears more like an average