Heart disease detection

Date
Page

- from sklearn.naive_bayes import GaussianNB

$$P(y/X) = \frac{P(X/y) \cdot P(y)}{P(X)}$$

Gaussian Naive Bayes = Simple, fast algorithm based on Bayes' theorem

"Gaussian" = assumes features are normally distributed (bell curve shape)

BAYES THEOREM: rule in probability theory that lets us update our brief about a condition when we get new evidence.

P(A/B) = Probability of A given B

- from sklearn.ensemble import RandomForestClassifier

Random forest = Many Decision trees + Voting.

P(B/A) = Probability of B given A
P(A) = probability of A (prior)
P(B) = " " B (evidence)

$$P(\text{class}/\text{features}) = \frac{P(\text{features}/\text{class}) \cdot P(\text{class})}{P(\text{features})}$$

- from sklearn.ensemble import GradientBoostingClassifier.

While Random forest builds multiple trees in parallel & combines their votes, Gradient Boosting builds them one at a time, each one trying to fix the errors of the previous one.

1. Starting with a dumb model
2. Learn from mistakes
3. Add new correlation to previous prediction
4. Repeat

Final Model:

$$\hat{y}_{final} = \hat{y}_0 + \sum_{i=1}^{N} Tree_i(x)$$

y = true value
$\hat{y}_0$ = 1st weak prediction
$\hat{y}_1 = \hat{y}_0 + correction_1$
$\hat{y}_2 = \hat{y}_1 + correction_2$

- from sklearn.preprocessing import StandardScaler

Standardizes the features (i.e. makes all columns have • mean = 0 • Std dev = 1)

↳ some models like KNN & SVM are distance based (i.e get confused if one feature dominates others)

- from sklearn.neighbors import KNeighborsClassifier

looks at nearby points to decide the class of new data point.

- from sklearn.svm import SVC

this is to find the best boundary that maximally seperates the classes.