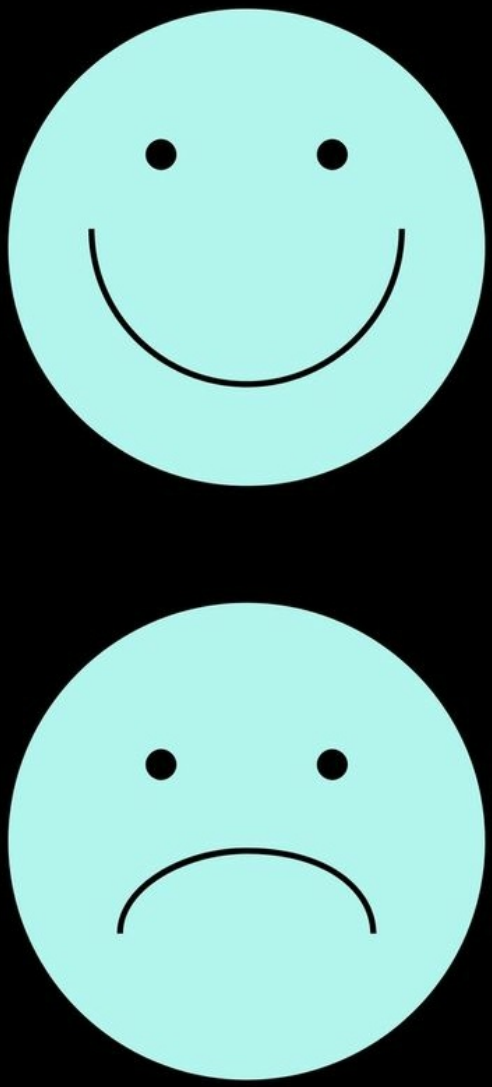
BEST & WORST CASES OF MACHINE LEARNING ALGORITHMS



Linear regression algorithm

Best cases

- It is good for interpretability and feature importance using weights (coefficients).
- If Feature engineering is done then this model can work better.

Worst cases

- Outliers impact a lot.
- When multicollinearity exists these models will not work properly.

Logistic regression algorithm

Best cases

- It works best when data is almost linearly separable and it is good if you have a low latency requirement
- It is good for interpretability and feature importance using weights (coefficients).
- Less impact of outliers because of the sigmoid.
- If dimensionality is large it workS well.

Worst cases

- It works badly when data is not linearly separable
- When data is imbalanced
- missing values.
- No multi-class classification for the base model.
- When multicollinearity exists these models will not work properly.

Naive Bayes algorithm

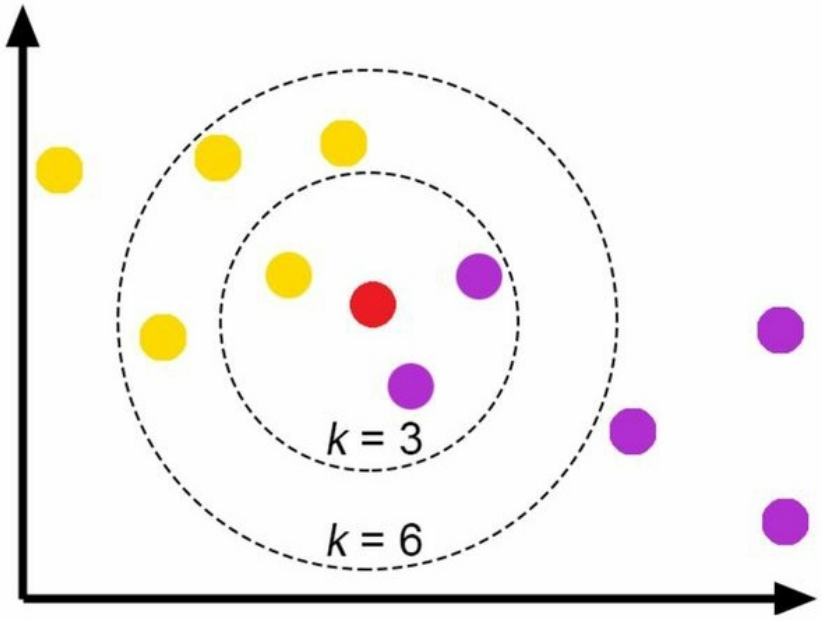
Best cases

- If the Conditional independence assumption of naive Bayes is true then it performs very well.
- Naive Bayes is the default algorithm when solving text classification problems.
- Naive Bayes is often used when you have categorical features (binary)
- Great interpretability, feature importance, low run time

Worst cases

- If the Conditional independence assumption of naive Bayes is false then its performance deteriorates
- Naive Bayes is not often used when you have real value features.
- Easily get's overfitted when Laplace smoothing is not done correctly.

KNN algorithm



Best cases

- If dimensionality (no of features) is low then this works best.
- If you know the right distance measure then KNN is a good option.

Worst cases

- If dimensionality (no of features) is large then this may face the curse of dimensionality problem, large run time and less interpretability
- When you want a low latency product then KNN is not a good option.

Decision tree algorithm

Best cases

- Multi-class classification is possible.
- Interpretability and feature importance

Worst cases

- Imbalanced data impacts a lot.
- If dimensionality is large then training time is high.
- If you use one-hot encoding then training time will be high.
- Outliers will impact the model.

SVM algorithm

Best cases

- If you can find the right kernel then thins works at its best.
- This can be applied to non-linear problems.
- Interpretability and feature importance is easy for linear SVM's.
- The impact of outliers is less.
- If dimensionality is large then SVM works like a charm.

Worst cases

- Interpretability and feature importance is hard for kernel SVM's
- If training data is large, training time is high.