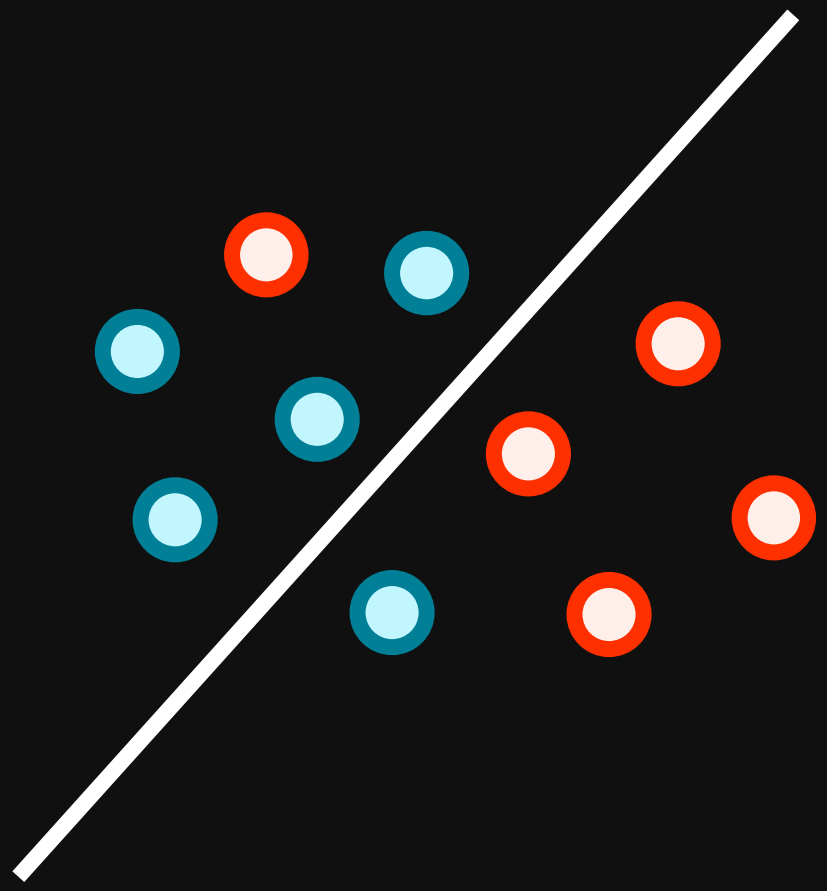
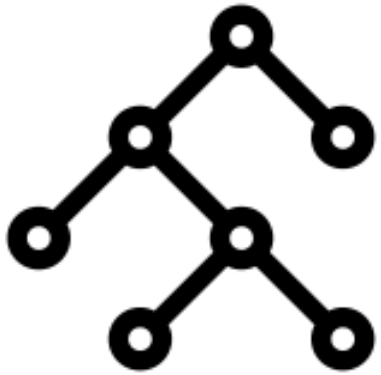
Basics of Machine Learning

Dmitry Ryabokon, github.com/dryabokon



Lesson 11
Non-parametrical ML methods

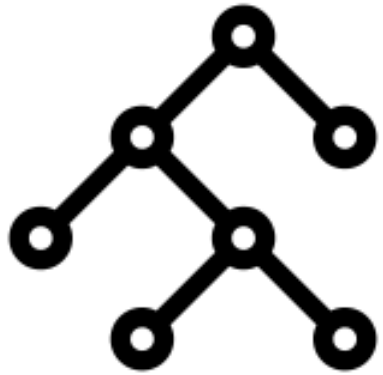


Supervised Learning

Summary

- Decision tree
- KNN
- Bias vs Variance tradeoff

Decision Tree



Decision tree

F1	F2	F3	Target
A	A	A	0
A	A	B	0
B	B	A	1
A	B	B	1

H(0,0,1,1) = ½ log(½) + ½ log(½) = 1

Decision tree

F1	F2	F3	Target	Attempt split on F1
A	A	A	0	0
A	A	B	0	0
B	B	A	1	1
A	B	B	1	0

Information Gain = H(0011) - 3/4 H(001) - 1/4 H(1) = 0.91

Decision tree

F1	F2	F3	Target	Attempt split on F2
A	A	A	0	0
A	A	B	0	0
B	B	A	1	1
A	B	B	1	1

Information Gain = H(0011) - 1/2 H(00) - 1/2 H(11) = 1

Decision tree

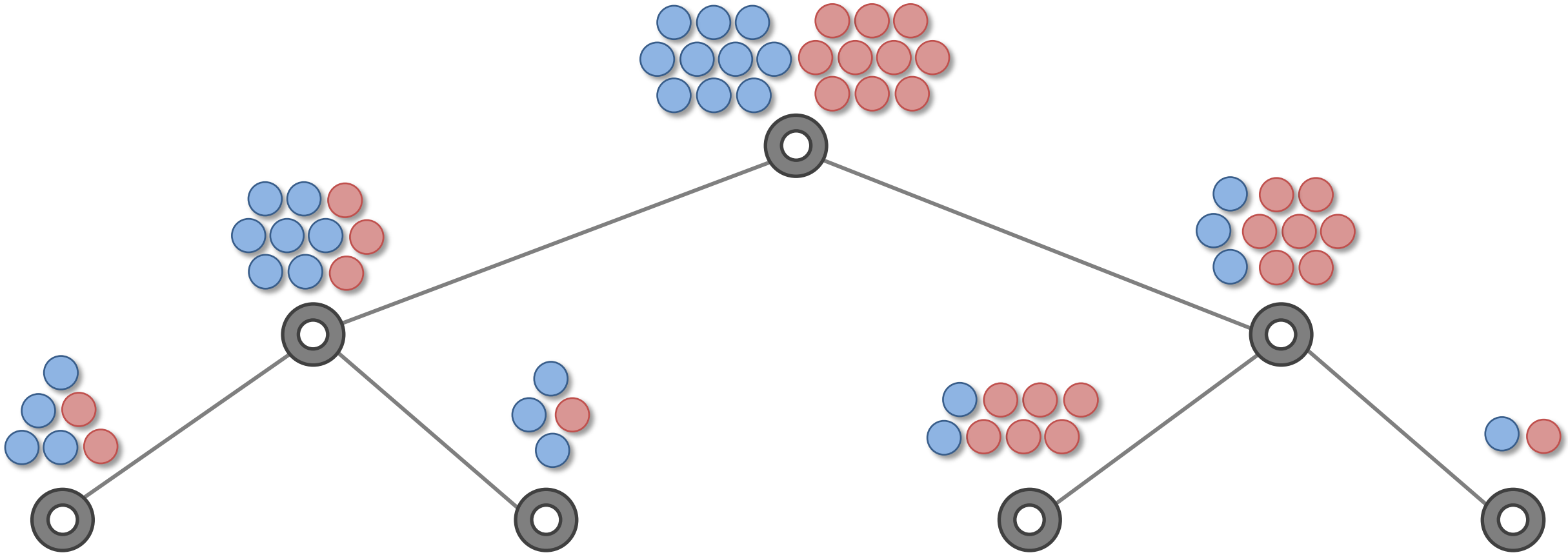
F1	F2	F3	Target	Attempt split on F3
A	A	A	0	0
A	A	B	0	0
B	B	A	1	1
A	B	B	1	1

Information Gain = H(0011) - 1/2 H(01) - 1/2 H(01) = 0

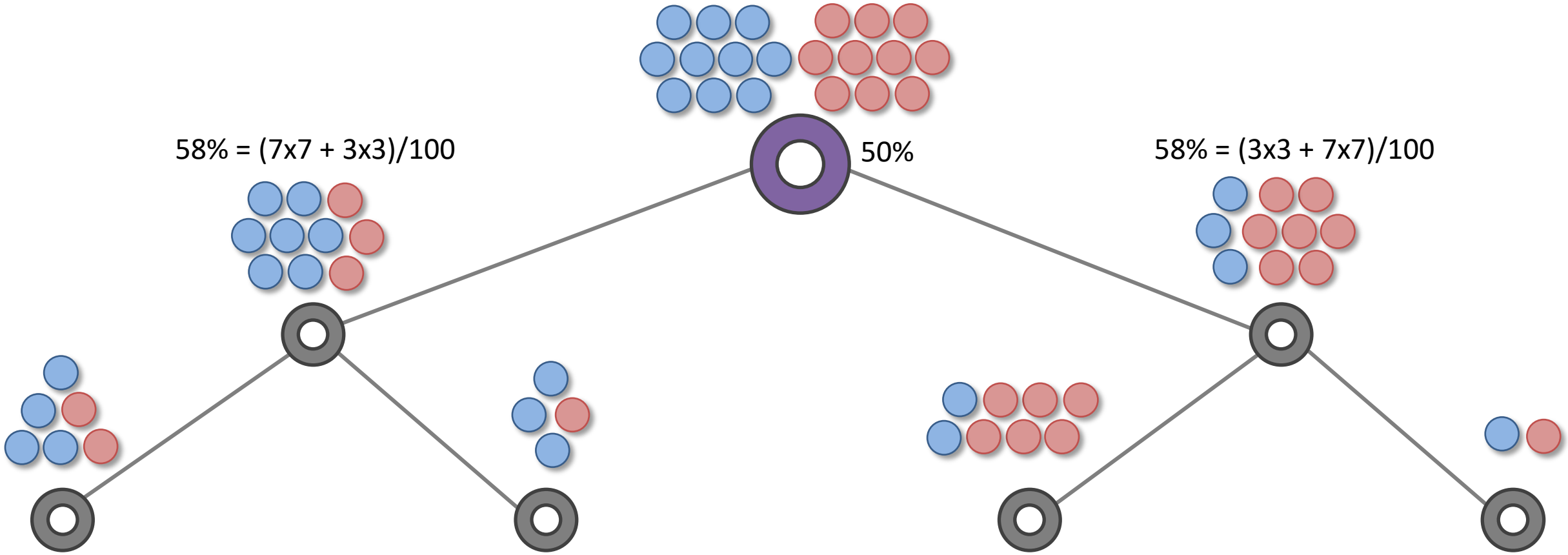
Decision tree

F1	F2	F3	Target
A	A	A	0
A	A	B	0
B	B	A	1
A	B	B	1

Decision tree: pruning



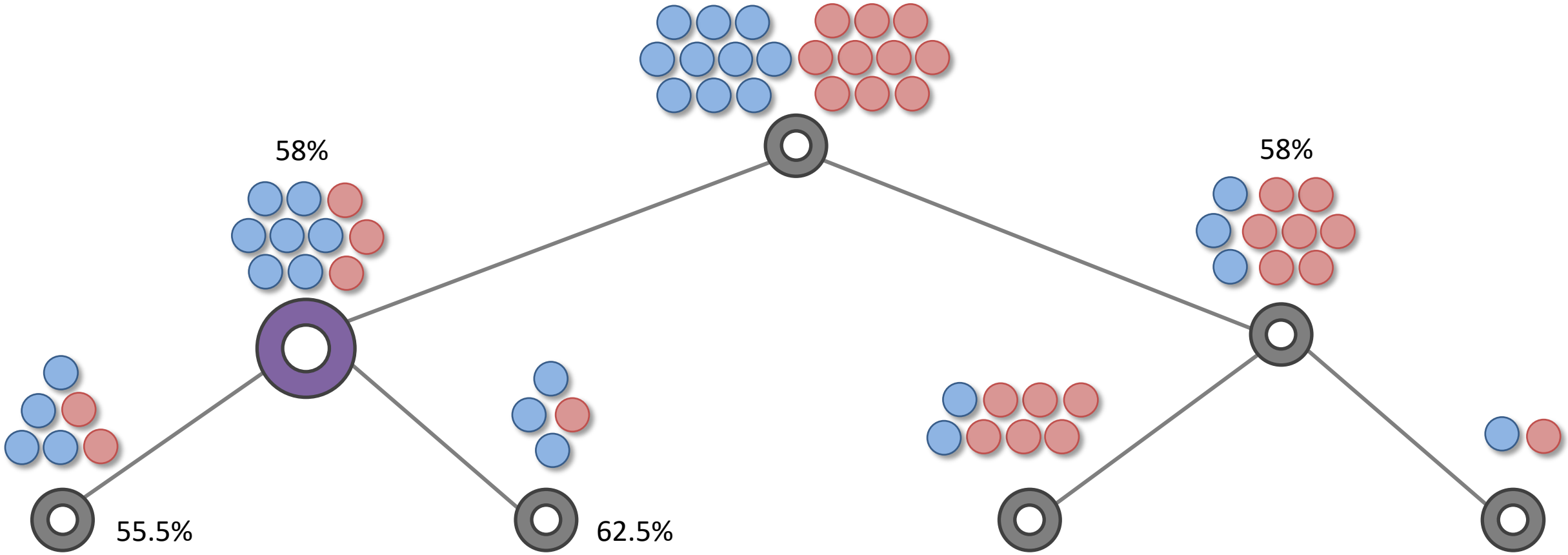
Decision tree: pruning



50% x 58% + 50% x 58% = 58%

58% > 50% OK

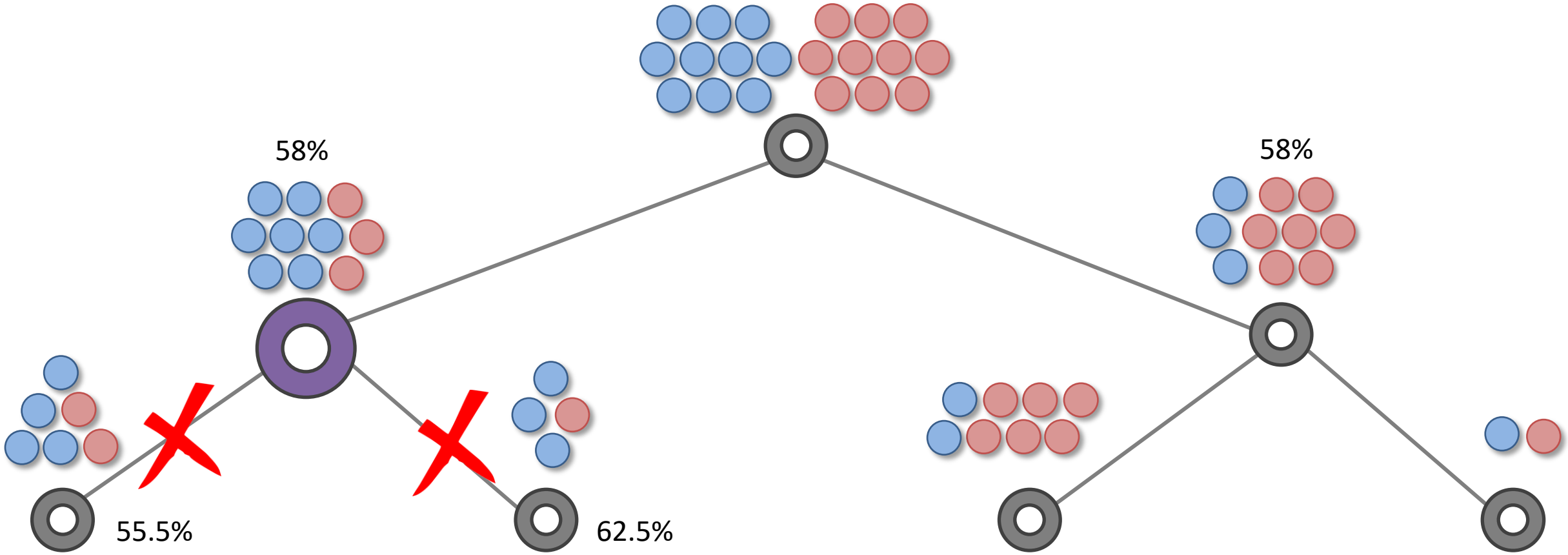
Decision tree: pruning



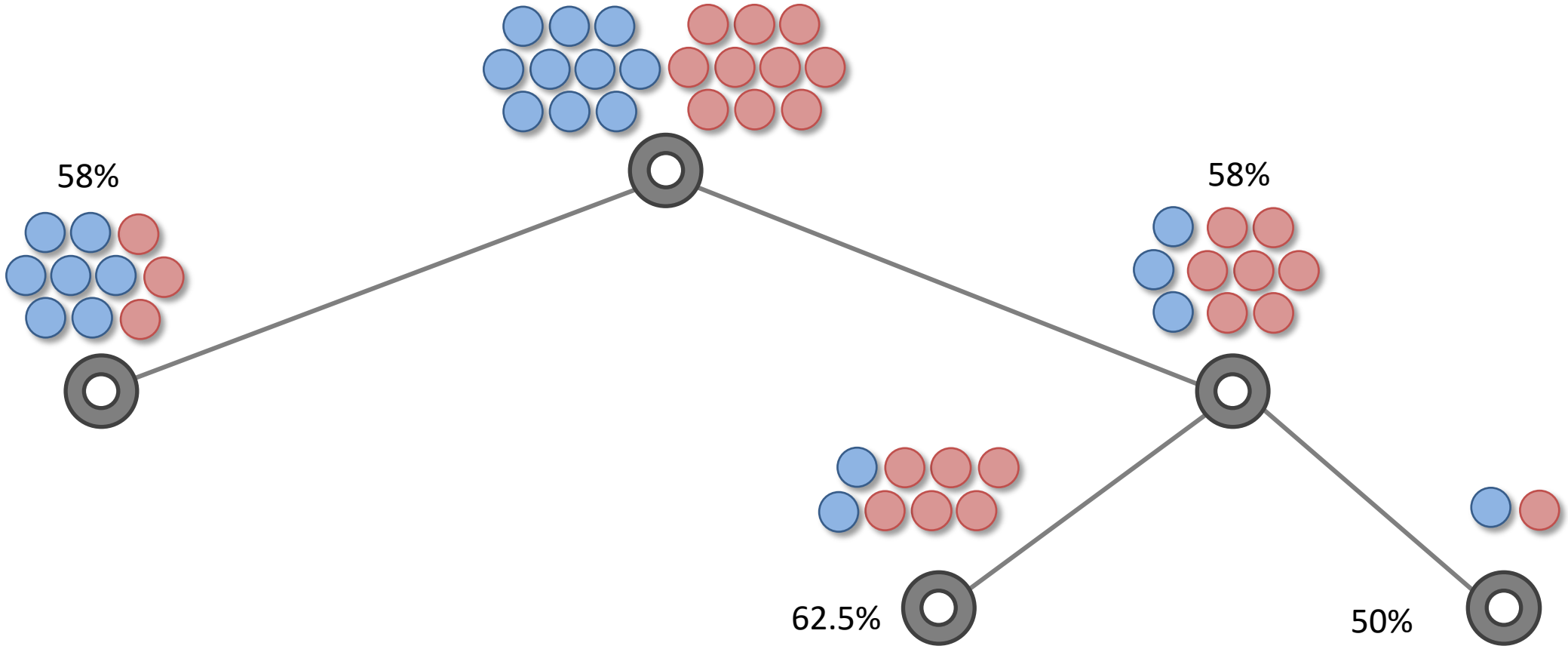
50% x 58% + 50% (60% x 55.5% + 40% x 62.5%) = 50% x 58% + 50% x 58.3 = 58.16%

58.16% > 58% *this is small improvement*

Decision tree: pruning



Decision tree: pruning

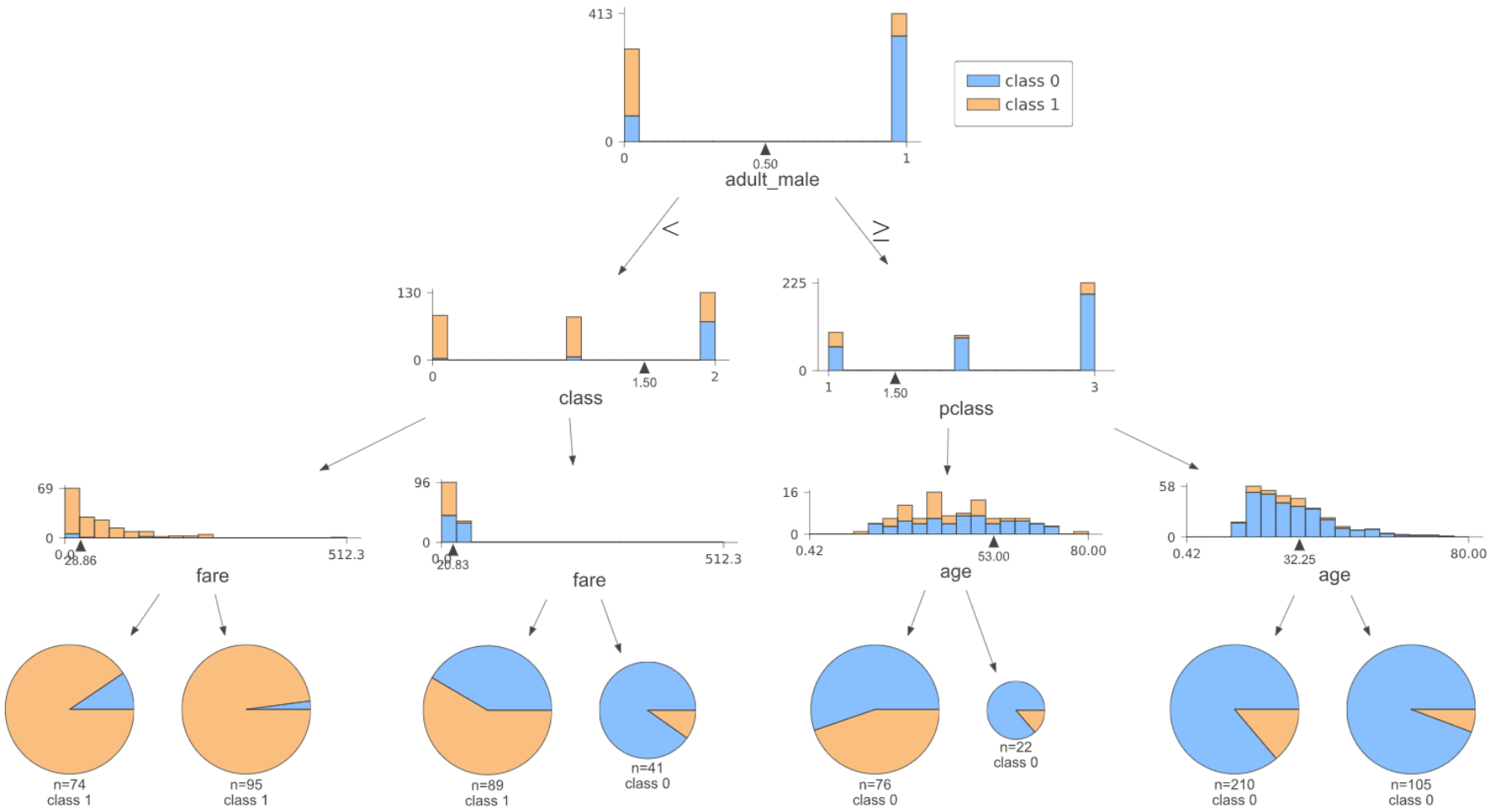


50% x 58% + 50% (80% x 62.5% + 20% x 50%) = 50% x 58% + 50% x 60% = 59%

59% > 58% *OK*

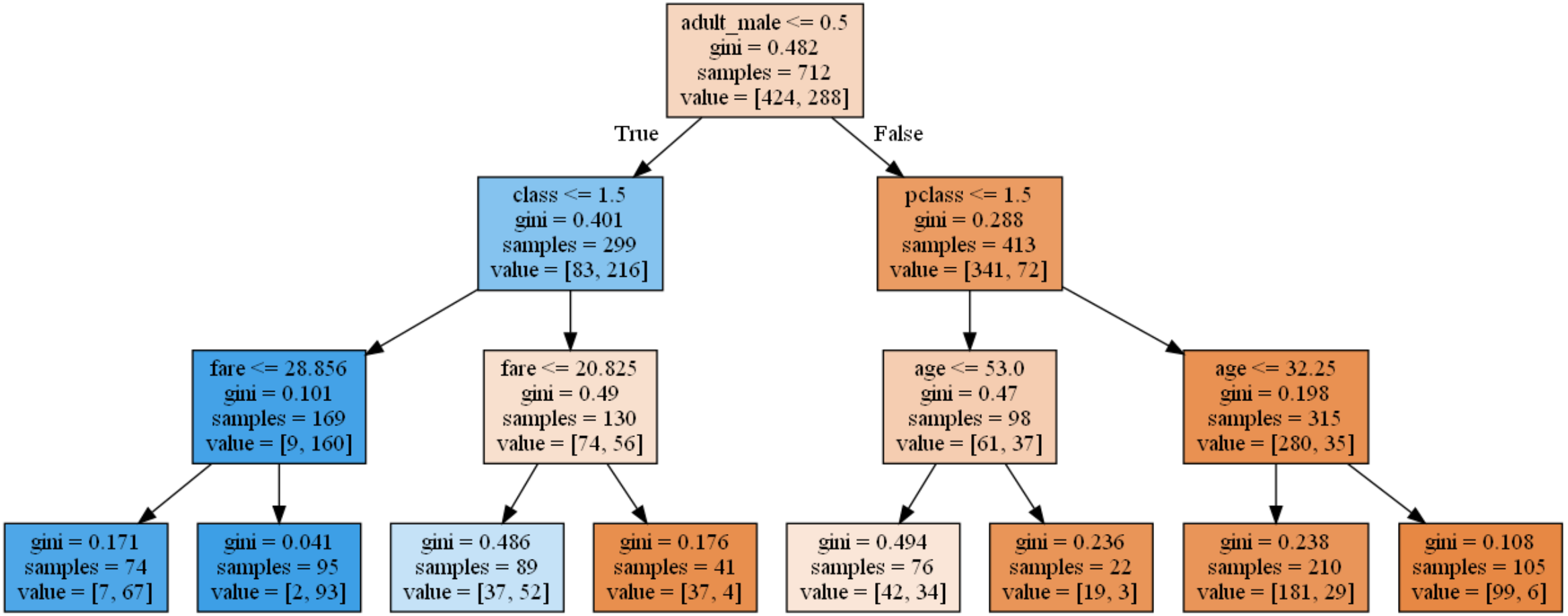
Decision tree: pruning

Example: Titanic



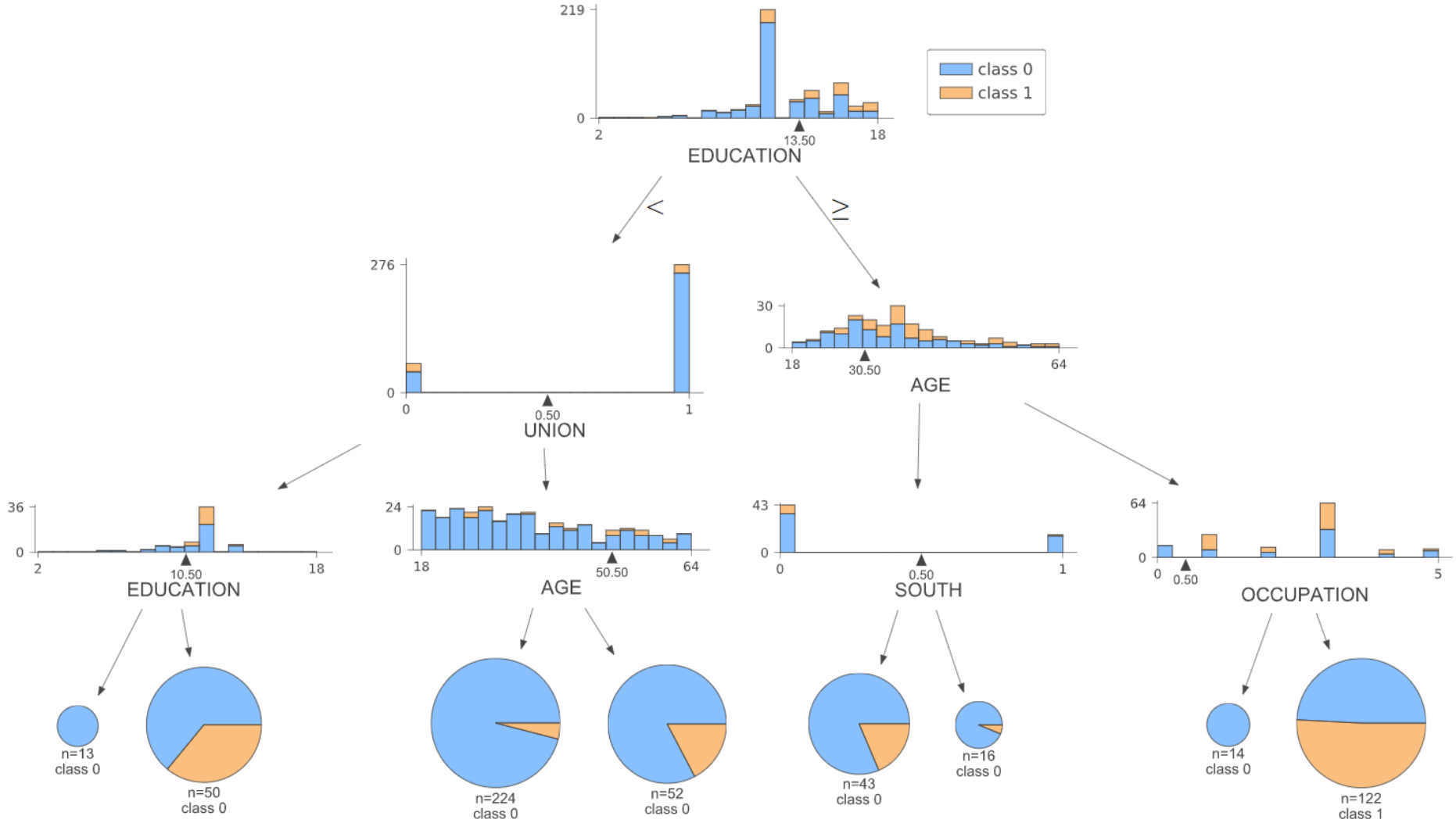
Decision tree: pruning

Example: Titanic

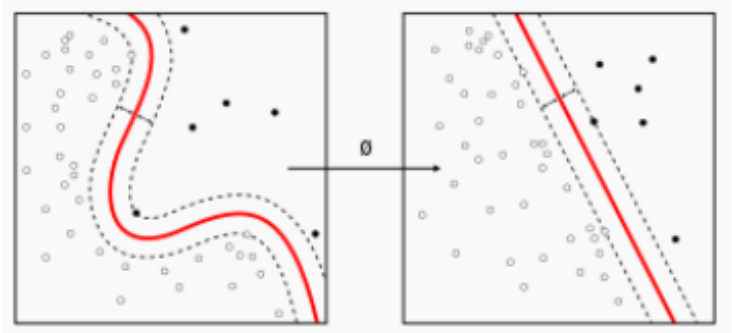


Decision tree: pruning

Example: wages



Bias vs Variance tradeoff



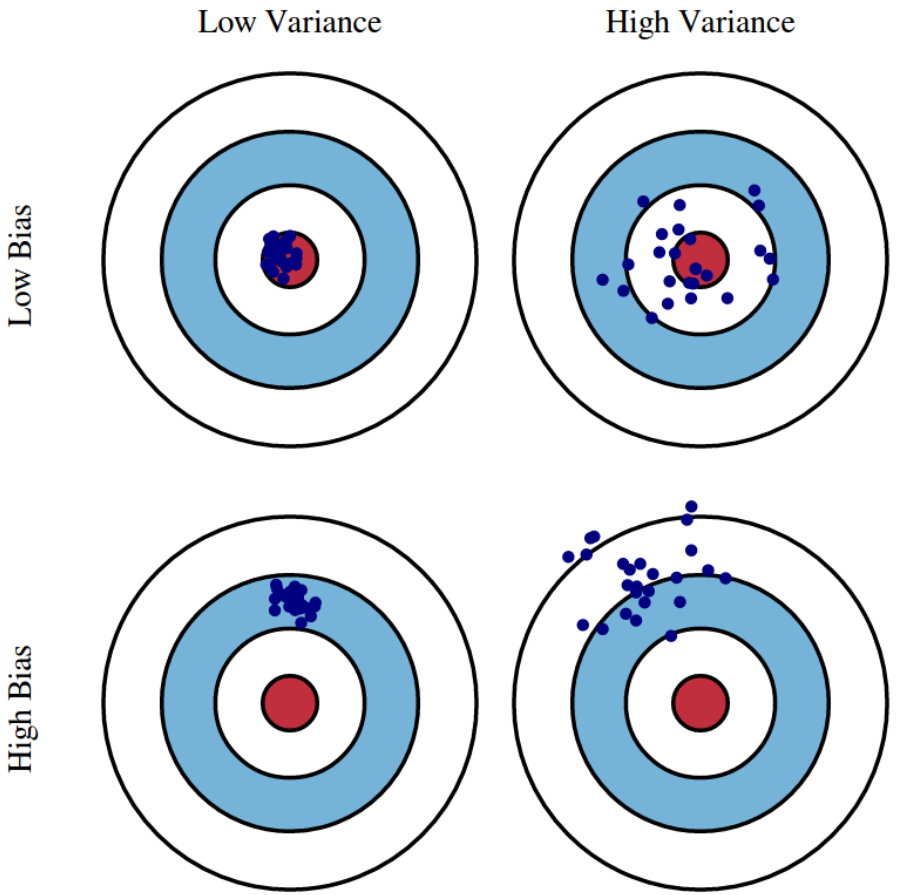
https://www.cs.cmu.edu/~wcohen/10-601/bias-variance.pdf

Bias vs Variance

Definitions

- **Bias** is an error from wrong assumptions in the learning algorithm
- High bias can cause **underfit**: the algorithm can miss relations between features and target

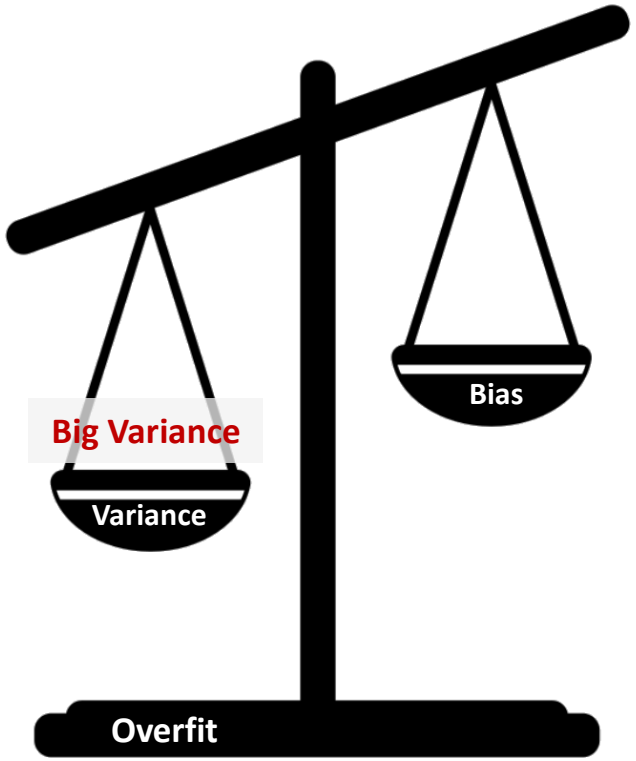
- The **variance** is an error from sensitivity to small fluctuations in the training set.
- High variance can cause **overfit**: the algorithm can model the random noise in the training data, rather than the intended outputs



https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html

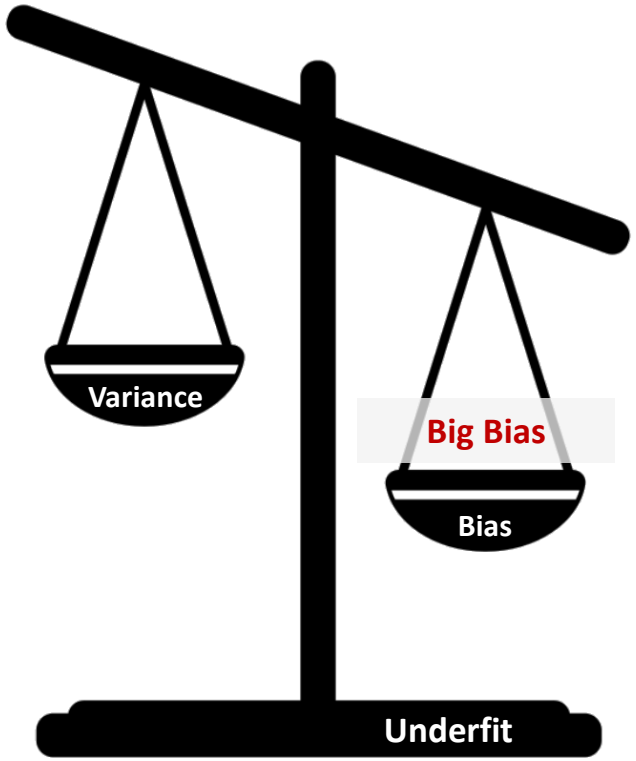
Bias vs Variance

Definitions



Bias vs Variance

Definitions



Bias vs Variance

Decomposition

- Lets imagine there are N possible training datasets $\{D_1, D_2, \ldots, D_N\}$
- For a given dataset one gets an estimators $g^{(D)}(x)$
- Lets denote the expected estimator by $\mathbf{E}_D\left[g^{(D)}(x)\right] \equiv \bar{g}(x)$
- If N is large we can approximate it by an average over all datasets (law of large numbers)

$$\bar{g}(x) \approx \frac{1}{N}\sum_{i=1}^{N} g^{(D_i)}(x)$$

- The **variance** of an estimator tells us how far particular predictions are from the mean valu

$$var = \mathbf{E}_D\left[\left(g^{(D)}(x) - \bar{g}(x)\right)^2\right]$$

- Thus, if the training does not depend on the choice of a dataset the variance is low
- The **bias** of an estimator tells us how far the mean value is from the true value

$$bias = \bar{g}(x) - f(x)$$

- Lets consider MSE for a particlar point x, $y = f(x) + \epsilon$, so

$$mse = \mathbf{E}_D\left[\left(g^{(D)}(x) - y\right)^2\right] = \mathbf{E}_D\left[\left(g^{(D)}(x)\right)^2\right] - 2\cdot\mathbf{E}_D\left[g^{(D)}(x)\cdot y\right] + \mathbf{E}_D\left[y^2\right]$$

- Here, we used the linearity of the expected value operator. Lets use another common property: $\mathbf{E}\left[X^2\right] = \mathbf{E}\left[\left(X - \mathbf{E}\left[X\right]\right)^2\right] + \mathbf{E}[X]^2$
- So the first term can be rewritten in the form

$$\mathbf{E}_D\left[\left(g^{(D)}(x)\right)^2\right] = \mathbf{E}_D\left[\left(g^{(D)}(x) - \mathbf{E}_D\left[g^{(D)}(x)\right]\right)^2\right] + \mathbf{E}_D\left[g^{(D)}(x)\right]^2 = \mathbf{E}_D\left[\left(g^{(D)}(x) - \bar{g}(x)\right)^2\right] + \left(\bar{g}(x)\right)^2$$

- And the last term

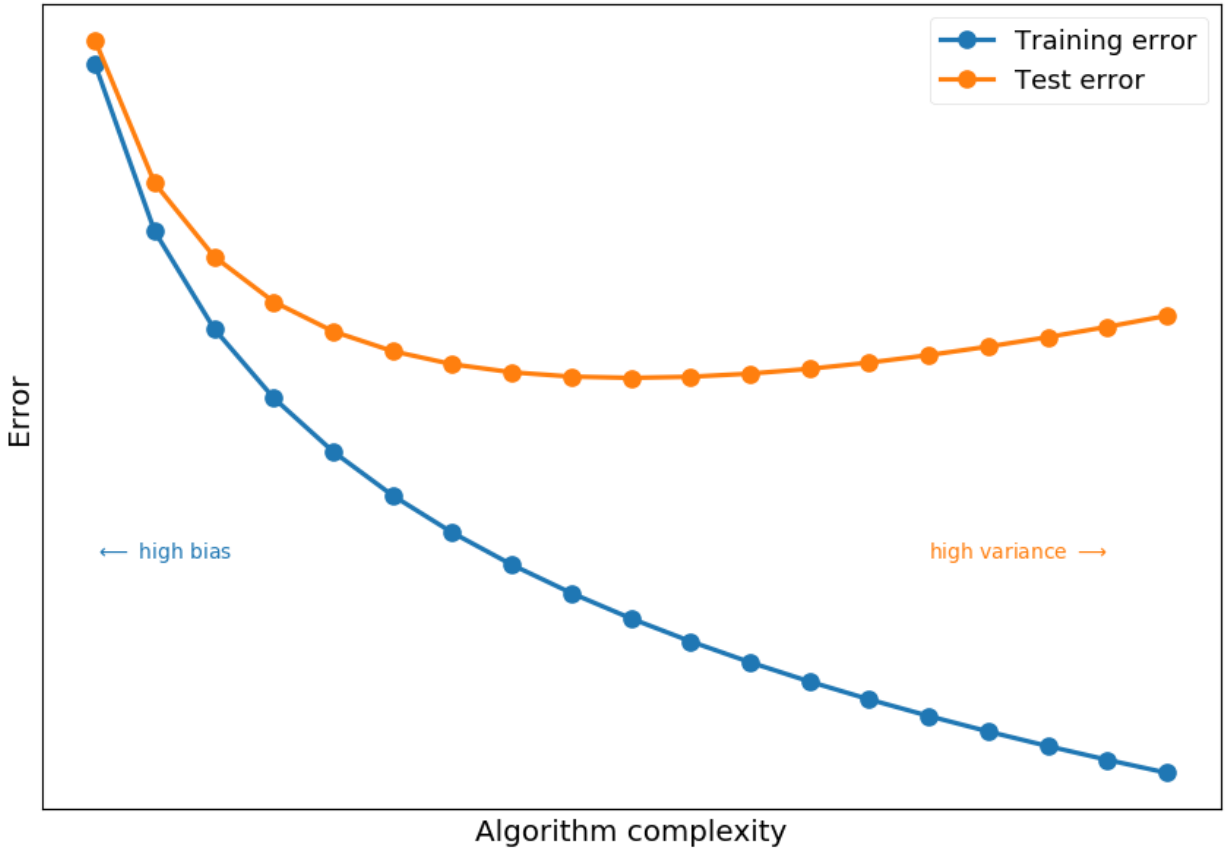
$$\mathbf{E}_D\left[y^2\right] = \mathbf{E}_D\left[\left(y - \mathbf{E}_D\left[y\right]\right)^2\right] + \mathbf{E}_D[y]^2 = \mathbf{E}_D\left[\left(y - f(x)\right)^2\right] + \left(f(x)\right)^2$$

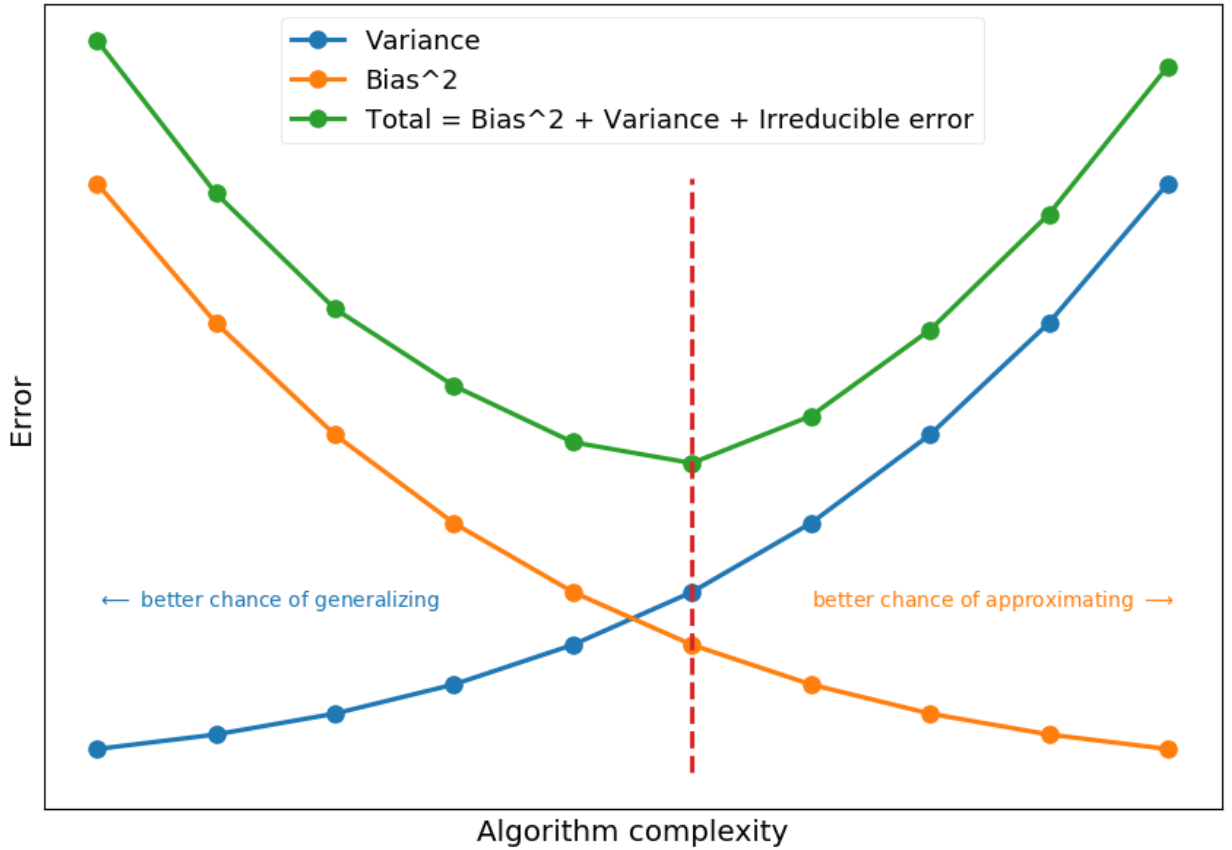
- Here, we used the fact that $\mathbf{E}_D\left[y\right] = f(x)$ (noise would average out when averaging over *infinite* number of datasets)
- For the middle term we use the fact that for independent X and Y: $\mathbf{E}\left[XY\right] = \mathbf{E}\left[X\right]\cdot\mathbf{E}\left[Y\right]$, so $\mathbf{E}_D\left[g^{(D)}(x)\cdot y\right] = \bar{g}(x)\cdot f(x)$
- Taking all together we get

$$mse = \underbrace{\mathbf{E}_D\left[\left(g^{(D)}(x) - \bar{g}(x)\right)^2\right]}_{variance} + \underbrace{\left(\bar{g}(x) - f(x)\right)^2}_{bias^2} + \underbrace{\mathbf{E}_D\left[\left(y - f(x)\right)^2\right]}_{noise}$$

Bias vs Variance

Model complexity





https://tomaszgolan.github.io/introduction_to_machine_learning/markdown/introduction_to_machine_learning_02_dt/introduction_to_machine_learning_02_dt/

Bias vs Variance

How to avoid overfit (high variance)

- Keep the model simpler, reduce model complexity
- Do cross-validation (e.g. k-folds)
- Use regularization techniques (e.g. LASSO, Ridge)
- Use top n features from variable importance chart
- Use bagging

Bias vs Variance

How to avoid underfit (high bias)

- Make model more complex
- Add features
- Use boosting

Bias vs Variance

Regularization: when # of variables > # of observation

- In such high dimensional data sets, we can't use classical regression techniques, since their assumptions tend to fail. When p > n, we can no longer calculate a unique least square coefficient estimate, the variances become infinite, so OLS (Ordinary Least Squares) cannot be used at all.

- To combat this situation, we can use penalized regression methods like lasso, LARS, ridge which can shrink the coefficients to reduce variance. Precisely, ridge regression works best in situations where the least square estimates have higher variance.

Regularization: ridge (L2) and lasso (L1)

Regularization becomes necessary when the model begins to ovefit / underfit. This technique introduces a cost term for bringing in more features with the objective function. Hence, it tries to push the coefficients for many variables to zero and hence reduce cost term. This helps to reduce model complexity so that the model can become better at predicting (generalizing).

The **key difference** between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for **feature selection** in case we have a huge number of features. Traditional methods like cross-validation, stepwise regression to handle overfitting and perform feature selection work well with a small set of features but these techniques are a great alternative when we are dealing with a large set of features.

In presence of many variables with small / medium sized effect, use ridge regression. Conceptually, we can say, lasso regression (L1) does both variable selection and parameter shrinkage, whereas Ridge regression only does parameter shrinkage and end up including all the coefficients in the model. In presence of correlated variables, ridge regression might be the preferred choice. Also, ridge regression works best in situations where the least square estimates have higher variance.

References

- https://mljar.com/blog/visualize-decision-tree/