Great post Richard, much appreciated! These issues have been on my mind recently as I’ve been playing around with fitting models to feature neutral targets. I’ve been testing out the Sortino ratio as an alternative to Sharpe for doing hyperparameter selection, because it makes sense to me to only penalize downside volatility/variance. Interestingly I’m finding that Sortino does favor different and narrower ranges of hyperparameters than Sharpe.

def sortino_ratio(x, target=.02): xt = x - target return np.mean(xt) / (np.sum(np.minimum(0, xt)**2)/(len(xt)-1))**.5

After reading your post and doing some internet searching I came across this document which proposes a modification to Sharpe, they call Smart Sharpe, which takes autocorrelation into account. If anyone is interested I threw together a simple implementation to help clarify it to myself. I also created the “Smart” version of Sortino by including the autocorrelation penalty term to perhaps get the best of both worlds.

keyquant.com

[

](https://www.keyquant.com/Download/GetFile?Filename=%5CPublications%5CKeyQuant_WhitePaper_APT_Part2.pdf)

GetFile

2.02 MB

def ar1(x): return np.corrcoef(x[:-1], x[1:])[0,1]

def autocorr_penalty(x): n = len(x) p = ar1(x) return np.sqrt(1 + 2*np.sum([((n - i)/n)*p**i for i in range(1,n)]))

def smart_sharpe(x): return np.mean(x)/(np.std(x, ddof=1)*autocorr_penalty(x))

def smart_sortino_ratio(x, target=.02): xt = x - target return np.mean(xt)/(((np.sum(np.minimum(0, xt)**2)/(len(xt)-1))**.5)*autocorr_penalty(x))