UDACITY

Return to Classroom

Multifactor Model

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Congratulations

Great project! Hope you have enjoyed this course and are highly enthusiastic to gain more knowledge and use your skills to go out in the real world. You can visit WorldQuantVRC which provides a simulator for free for everyone to test their strategies on their data sets (which include fundamentals, price-volume, news, sentiments, analyst ratings, etc). They also have tons of video lectures and links to research papers which you access for free.
Remember your job as a quant would be to hunt for **alpha factors** which will make you very rich in turn ;-)

Extra Materials

1. How to Calculate Principal Component Analysis (PCA) from Scratch in Python
2. A step by step tutorial to Principal Component Analysis.
3. What is the difference between LDA and PCA for dimensionality reduction:
4. Dimension Reduction Techniques (PCA vs LDA) in Machine Learning

5. Machine Learning FAQ: Difference between LDA and PCA
   What is the Difference Between a Sharpe Ratio and a Traynor Ratio?

Statistical Risk Model

The function `fit_pca` fits the PCA model with returns.

Great job fitting PCA. Remember PCA is unsupervised learning here so we are going to learn the variances and then see what features play a significant role in our risk model.

PCA In Depth: Principal Component Analysis

The function `factor_betas` gets the factor betas from the PCA model.

Similar implementation!

Easy for others to understand if you plan to put it up on Github!
Like which one is column and which one is indices.I know you have named the variables like that but it is a good habit to specify the columns and index so that when other developer perceives at just one sight.

```
data = pca.components_.T
    return pd.DataFrame(data=data, columns=factor_beta_columns, index=factor_beta_indices)
```

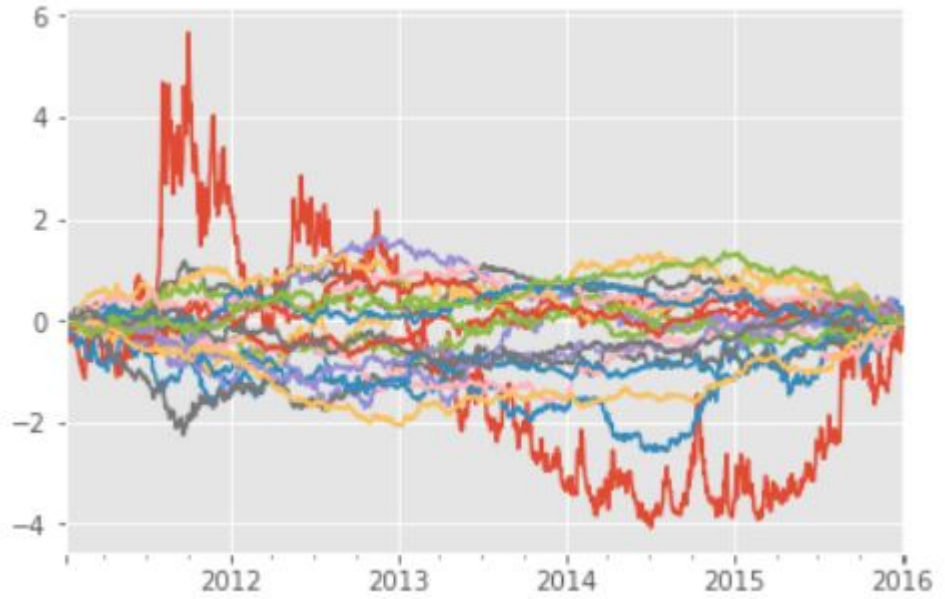
The function `factor_returns` gets the factor returns from the PCA model.

Awesome, you've correctly generated the factor returns from your PCA model.
Note:

Here the PCA with the transform method gets the (transformed) Factor Returns. These factor returns will be used to calculate the variance and covariance of the risk factors. Check out more details on the PCA's transform method in the sklearn here:

- Examples using sklearn.decomposition.PCA
- Understanding scikit learn pca transform function in python.

Here we can see that what these factor returns looks like over time,



The function `factor_cov_matrix` gets the factor covariance matrix.

Well done!
👏

Could not be a better implementation.

Covariance and Correlation

The function `idiosyncratic_var_matrix` gets the idiosyncratic variance matrix.

Well done!

Idiosyncrasies and challenges of data driven learning in electronic trading

---

The function `idiosyncratic_var_vector` gets the idiosyncratic variance vector.

The function `predict_portfolio_risk` gets the predicted portfolio risk.

Well done!

Correct and readable!

Create Alpha Factors

The function `mean_reversion_5day_sector_neutral` generates the mean reversion 5 day sector neutral factor.

Awesome !

Note you can just call `-momentum_1yr` as well!

The function `mean_reversion_5day_sector_neutral_smoothed` generates the mean reversion 5 day sector neutral smoothed factor.

Well done!

👏
🌟🌟🌟🌟🌟

Pay special attention to the `zscore()` operator since you would be using that a lot while designing alpha factors.

Evaluate Alpha Factors

The function `sharpe_ratio` gets the sharpe ratio for each factor for the entire period.

Fantastic, your sharpe_ratio function is correctly implemented.
Sharpe ratio is the measure of risk-adjusted return of a financial portfolio. A portfolio with a higher Sharpe ratio is considered superior relative to its peers.

You'll notice that the smoothed factors have a lower sharpe ratio.

Also checkout `np.var()` for an alternative implementation.

The student correctly mentions what would happened if you smooth the momentum factor and why.

You have concluded exactly right and this is the closest to what we were expecting from you but the thing is you cannot be sure about what would happen if we smooth the momentum factor just like you mentioned

> it would help and have more impact if the duration was shorter.
> This is the answer I expect from learners. Excellent Job! ❤️

It depends on the smoothing window: smoothing should improve the performance if the window is not too long, however, if it is long, it is hard to say what would happen.

Some very important facts which have to be remembered:

Smoothing: Smoothing means taking weighted average of portfolio weights over multiple days so as to reduce our turnover and thus our transaction costs. Our full service brokerage charges commissions (and not flat fees) and thus if we trade in and out a lot, we would spend a lot in brokerage charges. Thus even if we save 100 to 200 basis points (1%-2%) in transaction costs, we can easily make up our management fees of AUM which over the long run on $5 billion (WorldQuant's AUM) is a very significant amount.

Performance: We measure our risk adjusted returns as sharpe which takes volatility into consideration. The most important thing we have to remember here is that the performance is given if we had employed this strategy in the past and thus it does not guarantee future returns. We usually formulate alpha factors by some intuition and then frame the raw alpha. Then we try to perform fine tuning. We have to be very careful while fine tuning and shouldn't rely excessively on past performance since that will make our model to over-fit the data which we have leading to historical bias. Read more about this here: The Most Important Plot in Finance
To conclude, we smooth out to reduce our transaction costs irrespective of whether the sharpe increases or decreases.

Optimal Portfolio Constrained by Risk Model

The function `OptimalHoldings._get_obj` returns the correct objective function.

Good job, you generated the correct objective function.

The function `OptimalHoldings._get_constraints` returns the correct list of constraints.

The function `OptimalHoldingsRegualization._get_obj` returns the correct objective function.

Fantastic, you successfully generate the correct objective function for optimizing the returns with regularization.

The function `OptimalHoldingsStrictFactor._get_obj` returns the correct objective function.

Getting target_weights to match the shape of the weights is key in this function, you've correctly done that.
The objective function is correctly implemented as the L2 norm of the difference between weights and the target_weights.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START