A typical decision tree model has many rules, it intends to overfit. So, it is very intuitive for us to introduce the random forest model, which has several ways to deal with overfitting. Randomly picking rows, columns and voting with mode.

Let's say we are trying to choose the min\_samples\_leaf hyperparameter and want to avoid overfitting. How many training samples would we choose to be the minimum per leaf? In non-financial and non-time series machine learning, setting this hyperparameter is straightforward: you use grid search cross-validation to find the value that maximizes the model’s performance on validation data. When you have time-series data, you typically don’t use cross-validation because usually you just want a single validation dataset that is as close in time as possible to the present. If you have a problem with high signal-to-noise, then you can try a bit of parameter tuning on the single validation set.

In finance, though, you have time series data and you have low signal-to-noise. Therefore, you have one validation set and if you were to try a bunch of parameter values on this validation set, you would almost surely be overfitting. As such, you need to set the parameter with some judgement and minimal trials.

Some features may be not useful on its own, but when combined, they can become alpha. Conditional alphas.

一般来说，实例变量用于每个实例的唯一数据，而类变量用于类的所有实例共享的属性和方法:

**class** **Dog**:

kind = 'canine' *# class variable shared by all instances*

**def** \_\_init\_\_(self, name):

self.name = name *# instance variable unique to each instance*

Market regime could be told by market volatility and market dispersion. Market dispersion is calculated as the standard deviation of cross-sectional daily returns.

As the publication of alpha will surely decrease its profitability, could we take advantage of this sure thing?

Intuitively, if features 0 and 1 form the AND operator, then it makes sense that they should be equally important in determining the output. The feature importance calculated in sklearn assigns a higher importance to feature 0 compared to feature 1. This is because the tree first splits on feature 1, and then when it splits on feature 0, the labels become cleanly split into respective leaf nodes.

In other words, what we observe is that features which are used to split further down the bottom of the tree are given higher importance, using the Gini impurity as a measure.

**When feature importance measures are inconsistent**

There are many of methods for interpreting machine learning models, and for measuring feature importance. Many of these methods can be inconsistent, which means that the features that are most important may not always be given the highest feature importance score. We noticed this in the prior coding exercise, where there were two equally important features that form the “AND” operator, but one was given a feature importance of 0.33 because it was used for splitting the tree first, and the other was given a score of 0.67 because it was used for splitting second.

This is the motivation for using the latest feature attribution method, Shapley Additive Explanations, which we’ll see next.

If you wish to explore the concept of consistent feature attribution further, here’s a blog post that discusses some of the inconsistency seen in feature importance calculation methods. [Interpretable Machine Learning with XGBoost](https://towardsdatascience.com/interpretable-machine-learning-with-xgboost-9ec80d148d27). Understanding why a model makes a certain prediction can be as crucial as the prediction’s accuracy in many applications. However, the highest accuracy for large modern datasets is often achieved by complex models that even experts struggle to interpret, such as ensemble or deep learning models, creating a tension between accuracy and interpretability.

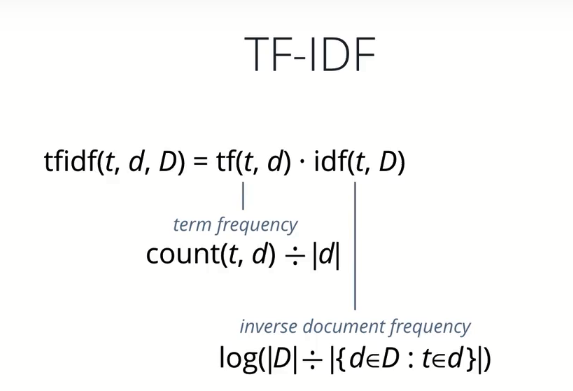
In response, various methods have recently been proposed to help users interpret the predictions of complex models, but it is often unclear how these methods are related and when one method is preferable over another. To address this problem, we present a unified framework for interpreting predictions, SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction. Its novel components include:

1. the identification of a new class of additive feature importance measures, and
2. theoretical results showing there is a unique solution in this class with a set of desirable properties. The new class unifies six existing methods, notable because several recent methods in the class lack the proposed desirable properties.

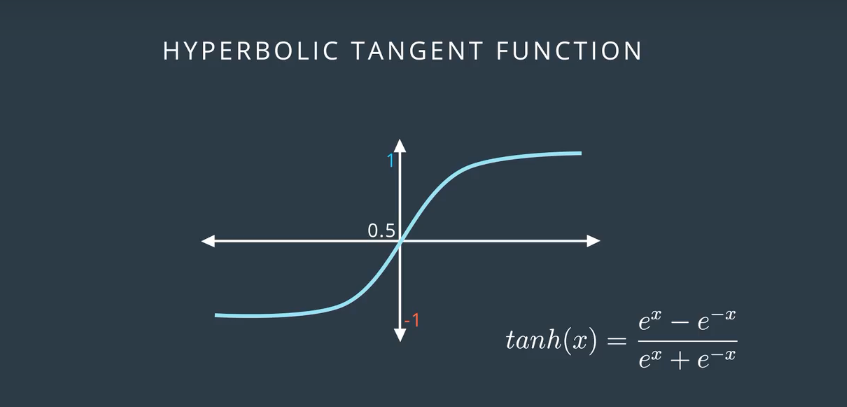
Based on insights from this unification, we present new methods that show improved computational performance and/or better consistency with human intuition than previous approaches.

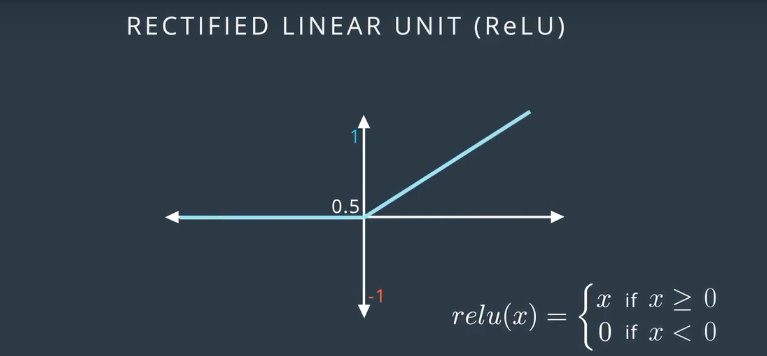
Accuracy vs interpretability

The creator of Shapley Additive Explanations, Scott Lundberg, has written an efficient implementation that we can install and use. We’ll be able to use this to determine both local feature importance (for a single observation) and global feature importance (for all training samples as a whole). To aggregate local feature importance into global feature importance, we take the absolute values of the local feature importances, and then average them.



Get 10-K and 10-Q from EDGAR—Electronic Data Gathering Analysis Retrieval





Deep learning structure

Define NN, criterion, optimizer, epochs, train

Train: loop epochs:

reset running\_loss

loop batches:

reset optimizer

load input

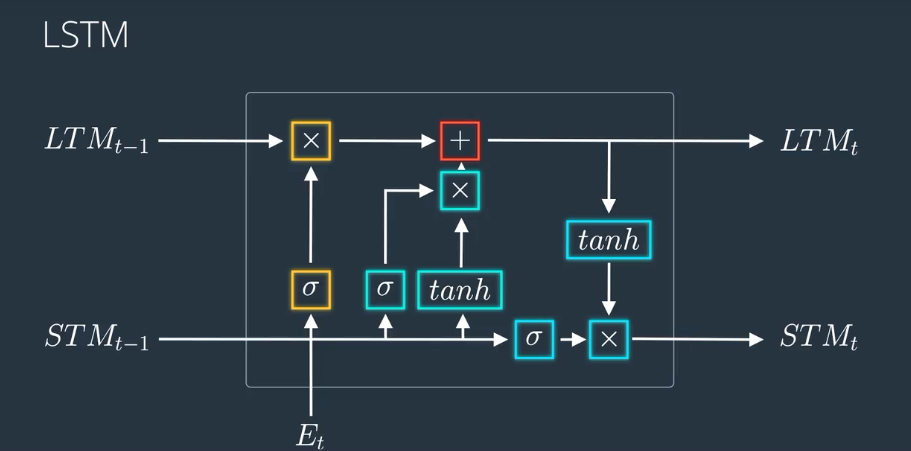
forward to get output

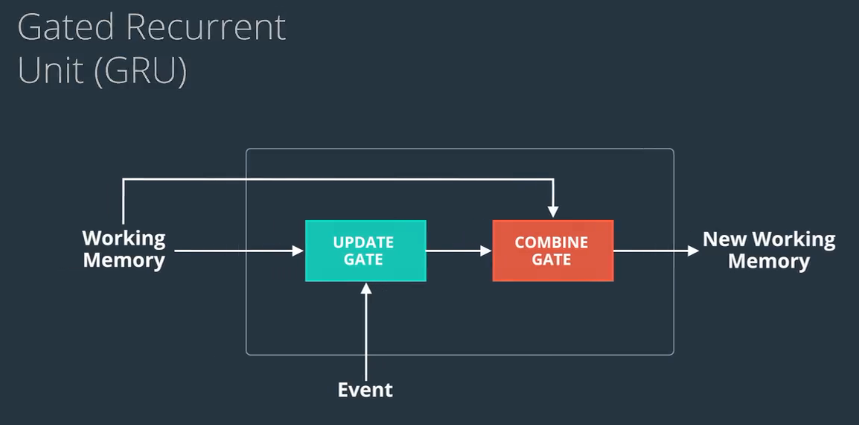
calculate loss

backward loss

update weights

update running\_loss





GRU is simpler but also works well in practice.

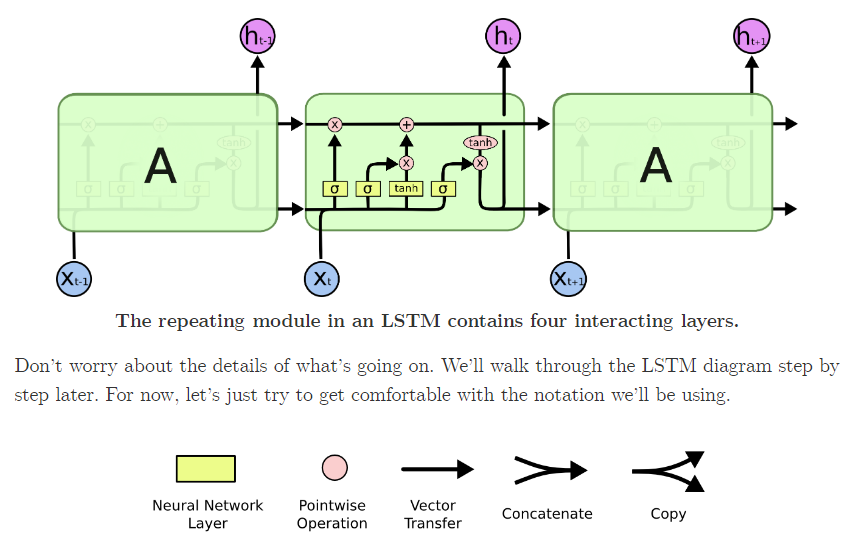
We need to clip the gradients (typically ReLU) to a threshold when perform LSTM. Because the gradients can explode easily here.

It is believed that simple models trained on huge amounts of data outperform complex systems trained on less data.

As we update the weights of the NN, do they keep the previous information?

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997) (Random Forest 1995, Ada boost 1996)and were refined and popularized by many people in following work. They work tremendously well on a large variety of problems and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!



3 Sigmoid, 3 multiply, 2 tanh, 1 sum

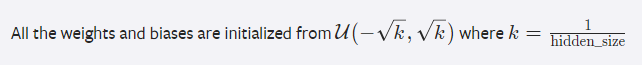
*If training vanilla neural nets is optimization over functions, training recurrent nets is optimization over programs.*

NN can simulate functions, while RNN can simulate programs.

它们二者是何其的相似，都把输出压缩在了一个范围之内。他们的导数图像也非常相近，我们可以从中观察到，sigmoid函数的导数范围是(0,0.25]，tanh函数的导数范围是(0,1]，他们的导数最大都不大于1。

这就会导致一个问题，在上面式子累乘的过程中，如果取sigmoid函数作为激活函数的话，那么必然是一堆小数在做乘法，结果就是越乘越小。随着时间序列的不断深入，小数的累乘就会导致梯度越来越小直到接近于0，这就是“梯度消失“现象。其实RNN的时间序列与深层神经网络很像，在较为深层的神经网络中使用sigmoid函数做激活函数也会导致反向传播时梯度消失，梯度消失就意味消失那一层的参数再也不更新，那么那一层隐层就变成了单纯的映射层，毫无意义了，所以在深层神经网络中，有时候多加神经元数量可能会比多家深度好。

dropout – If non-zero, introduces a Dropout layer on the outputs of each RNN layer except the last layer



net.eval() will set all the layers in your model to evaluation mode. This affects layers like dropout layers that turn "off" nodes during training with some probability but should allow every node to be "on" for evaluation. So, you should set your model to evaluation mode **before testing or validating your model**, and before, for example, sampling and making predictions about the likely next character in a given sequence. I'll set net.train() (training mode) only during the training loop.

If your training loss is much lower than validation loss, then this means the network might be **overfitting**. Solutions to this are to decrease your network size, or to increase dropout. For example, you could try dropout of 0.5 and so on.

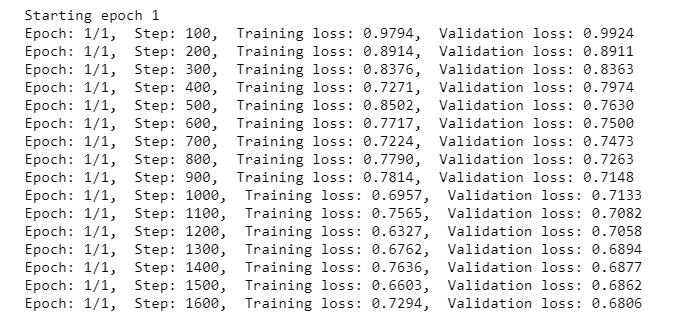
If your training/validation loss are about equal, then your model is **underfitting**. Increase the size of your model (either number of layers or the raw number of neurons per layer)

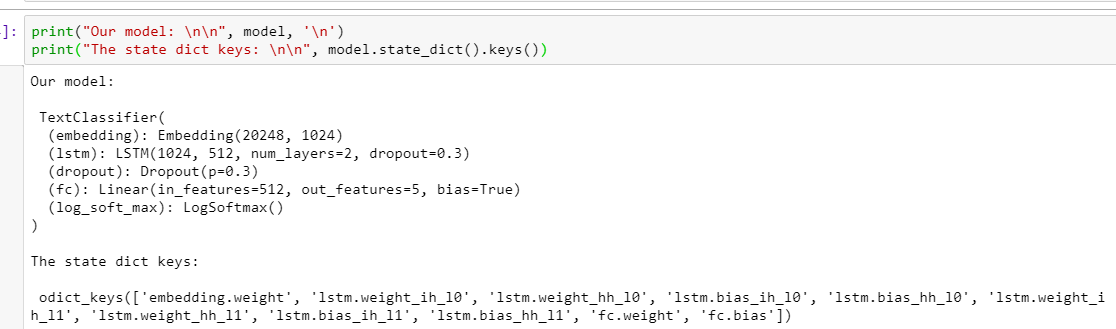
The two most important parameters that control the model are n\_hidden and n\_layers. I would advise that you always use n\_layers of either 2/3. The n\_hidden can be adjusted based on how much data you have. The two important quantities to keep track of here are:

* The number of parameters in your model. This is printed when you start training.
* The size of your dataset. 1MB file is approximately 1 million characters.

### Best models strategy

The winning strategy to obtaining very good models (if you have the compute time) is to always err on making the network larger (as large as you're willing to wait for it to compute) and then try different dropout values (between 0,1). Whatever model has the best validation performance (the loss, written in the checkpoint filename, low is good) is the one you should use in the end.





对比，Random Forest的子模型都拥有较低的偏差(rules 较多)，整体模型的训练过程旨在降低方差，故其需要较少的子模型（n\_estimators默认值为10）且子模型不为弱模型（max\_depth的默认值为None）；Gradient Tree Boosting的子模型都拥有较低的方差，整体模型的训练过程旨在降低偏差，故其需要较多的子模型（n\_estimators默认值为100）且子模型为弱模型（max\_depth的默认值为3）。

oob\_score ：默认识False，即是否采用袋外样本来评估模型的好坏。有放回采样中大约36.8% （1/e）的没有被采样到的数据，我们常常称之为袋外数据(Out Of Bag, 简称OOB)，这些数据没有参与训练集模型的拟合，因此可以用来检测模型的泛化能力。个人推荐设置为True，因为袋外分数反应了一个模型拟合后的泛化能力。对单个模型的参数训练，我们知道可以用cross validation（cv）来进行，但是特别消耗时间，而且对于随机森林这种情况也没有大的必要，所以就用这个数据对决策树模型进行验证，算是一个简单的交叉验证，性能消耗小，但是效果不错。

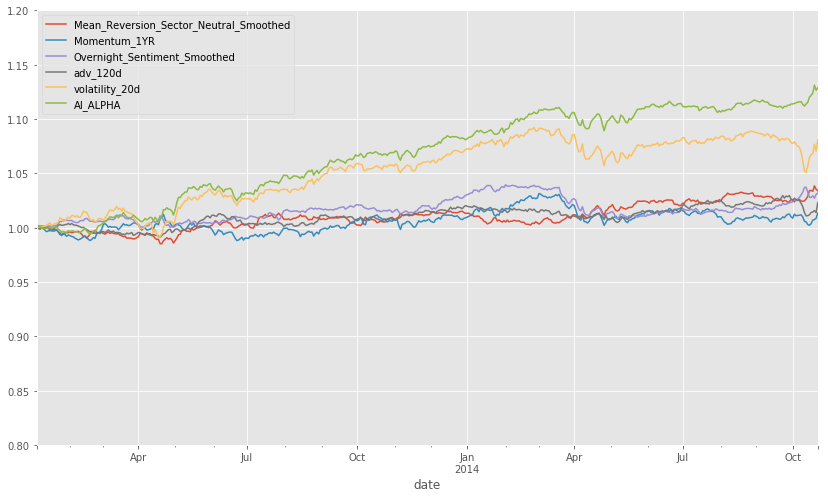
max\_features: RF划分时考虑的最大特征数。可以使用很多种类型的值，默认是"None",意味着划分时考虑所有的特征数；如果是"log2"意味着划分时最多考虑log2N个特征；如果是"sqrt"或者"auto"意味着划分时最多考虑N−−√N个特征。如果是整数，代表考虑的特征绝对数。如果是浮点数，代表考虑特征百分比，即考虑（百分比xN）取整后的特征数，其中N为样本总特征数。一般来说，如果样本特征数不多，比如小于50，我们用默认的"None"就可以了，如果特征数非常多，我们可以灵活使用刚才描述的其他取值来控制划分时考虑的最大特征数，以控制决策树的生成时间。

min\_samples\_split: 内部节点再划分所需最小样本数，默认2。这个值限制了子树继续划分的条件，如果某节点的样本数少于min\_samples\_split，则不会继续再尝试选择最优特征来进行划分。 默认是2.如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。

min\_samples\_leaf:叶子节点最少样本数。 这个值限制了叶子节点最少的样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。 默认是1,可以输入最少的样本数的整数，或者最少样本数占样本总数的百分比。如果样本量不大，不需要管这个值。如果样本量数量级非常大，则推荐增大这个值。

除了这些参数要注意以外，其他在调参时的注意点有：

1. 当样本少数量但是样本特征非常多的时候，决策树很容易过拟合，一般来说，样本数比特征数多一些会比较容易建立健壮的模型
2. 如果样本数量少但是样本特征非常多，在拟合决策树模型前，推荐先做维度规约，比如主成分分析（PCA），特征选择（Losso）或者独立成分分析（ICA）。这样特征的维度会大大减小。再来拟合决策树模型效果会好。
3. 推荐多用决策树的可视化（下节会讲），同时先限制决策树的深度（比如最多3层），这样可以先观察下生成的决策树里数据的初步拟合情况，然后再决定是否要增加深度。
4. 在训练模型先，注意观察样本的类别情况（主要指分类树），如果类别分布非常不均匀，就要考虑用class\_weight来限制模型过于偏向样本多的类别。
5. 决策树的数组使用的是numpy的float32类型，如果训练数据不是这样的格式，算法会先做copy再运行。
6. 如果输入的样本矩阵是稀疏的，推荐在拟合前调用csc\_matrix稀疏化，在预测前调用csr\_matrix稀疏化。



Sharpe Ratios

Mean\_Reversion\_Sector\_Neutral\_Smoothed 0.87000000

Momentum\_1YR 0.28000000

Overnight\_Sentiment\_Smoothed 0.83000000

adv\_120d 0.62000000

volatility\_20d 1.18000000

AI\_ALPHA 2.36000000

So, hopefully you are appropriately amazed by this. Despite the significant differences between the factor performances in the three sets, the AI APLHA can deliver positive performance.

2007/08 quant crisis

2008 financial crisis

#### Project 8: Backtesting

In this project, you will build a realistic backtester that uses the Barra data. The backtester will perform portfolio optimization that includes transaction costs, and you'll implement it with computational efficiency in mind, to allow for a reasonably fast backtest. You'll also use performance attribution to identify the major drivers of your portfolio's profit-and-loss (PnL). You will have the option to modify and customize the backtest as well.

#### Suggestion to customize your project

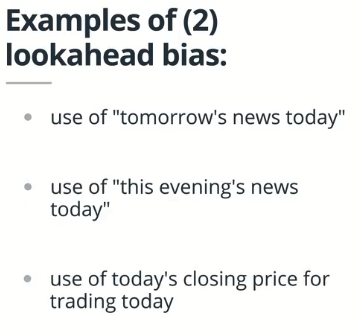
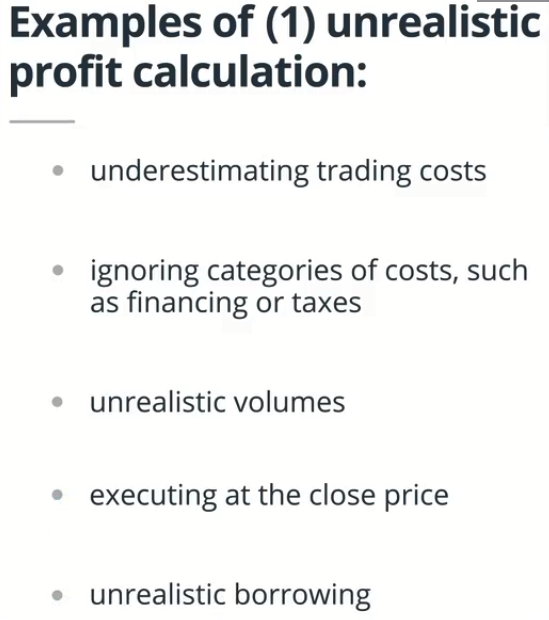
* Try backtesting on different time periods and interpret the final results.
* Try different factors to be their alphas.
* Try different weights for each alpha, based on some metric that tells us how confident we are in that alpha, such as a rolling average of the sharpe ratio for each alpha factor.
* Try different transaction cost models. Read the paper [**"Crossover from Linear to Square-Root Market Impact”.**](https://arxiv.org/pdf/1811.05230.pdf) It has a good overview of the transaction cost models, and it also references other papers that are useful in studying transaction cost models.
* Note about testing previous alphas: To test the alphas that you've created using the QuoteMedia data source, we would need a mapping file that identifies which cusip is associated with which barra ID. We currently aren't able to provide this in the classroom.

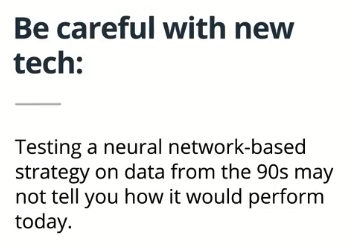
The curse of backtest.

Backtest is to optimize the performance in the future. So, don’t try to optimize the performance of the backtest. Backtest is only used to tell a strategy’s properties.

A valid backtest must

1. Trades and profits are achievable in the same way in the future.
2. No lookahead bias





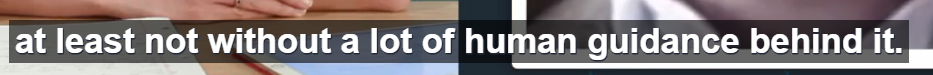
More competition may be faced.

1. Use cross-validation to achieve just the right amount of model complexity.
2. Always keep an out-of-sample test dataset. You should only look at the results of a test on this dataset once all model decisions have been made. If you let the results of this test influence decisions made about the model, you no longer have an estimate of generalization error.
3. Be wary of creating multiple model configurations. If the Sharpe ratio of a backtest is 2, but there are 10 model configurations, this is a kind of multiple comparison bias. This is different than repeatedly tweaking the parameters to get a Sharpe ratio of 2. ???
4. Be careful about your choice of time period for validation and testing. Be sure that the test period is not special in any way.
5. Keep track of the dates on which modifications to the model were made, so that you know the date on which a provable out-of-sample period commenced. If a model hasn’t changed for 3 years, then the performance on the past 3 years is a measure of out-of-sample performance.

Traditional ML is about fitting a model until it works. Finance is different—you can’t keep adjusting parameters to get a desired result. Maximizing the in-sample Sharpe ratio is not good—it would probably make out of sample Sharpe ratio worse. It’s very important to follow good research practices.

Focus on the coming year or two.





Every field needs prediction may also need AI’s power. It’s about automation and accuracy.

The assumption that we will use is that when the trade is 1% of Average Daily Volume, the price changes by 10 basis points. The transaction cost is the percent change in price times the amount traded. (or square root of adv)

When filling the risk factor matrix, we can use just the variances and set the covariances in the off-diagonals to zero. The covariance may be inaccurate. Including the covariance will be very computational costly. By only including variance will have to similar effect to having all the covariances.

How to optimize the objective function?

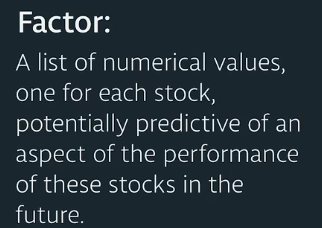
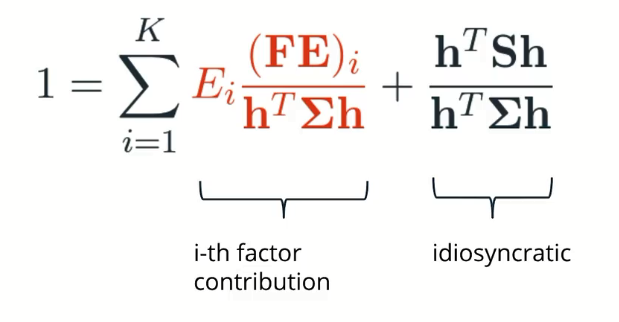
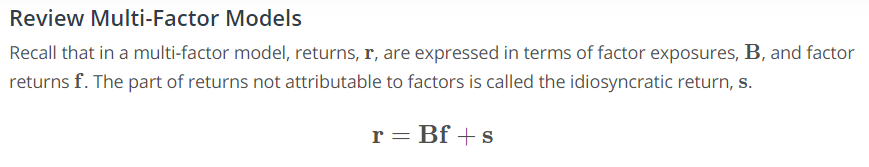
|  |  |
| --- | --- |
| **Without knowledge of the gradient:** | |
|  | * In general, prefer **BFGS** or **L-BFGS**, even if you have to approximate numerically gradients. These are also the default if you omit the parameter method - depending if the problem has constraints or bounds * On well-conditioned problems, **Powell** and **Nelder-Mead**, both gradient-free methods, work well in high dimension, but they collapse for ill-conditioned problems. | |
| **With knowledge of the gradient:** | |
|  | * **BFGS** or **L-BFGS**. * Computational overhead of BFGS is larger than that L-BFGS, itself larger than that of conjugate gradient. On the other side, BFGS usually needs less function evaluations than CG. Thus, conjugate gradient method is better than BFGS at optimizing computationally cheap functions. | |
| **With the Hessian:** | |
|  | * If you can compute the Hessian, prefer the Newton method (**Newton-CG** or **TCG**). | |
| **If you have noisy measurements:** | |
|  | * Use **Nelder-Mead** or **Powell**. | |

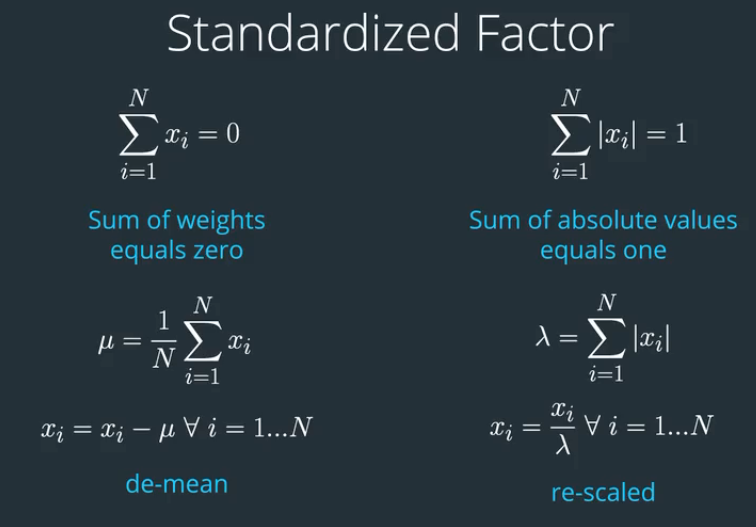
### [**Making your optimizer faster**](http://scipy-lectures.org/advanced/mathematical_optimization/#id45)

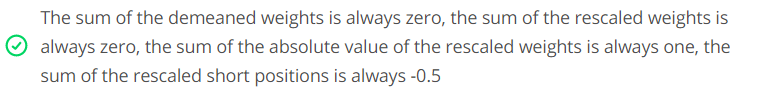
* Choose the right method (see above), do compute analytically the gradient and Hessian, if you can.
* Use [preconditionning](http://en.wikipedia.org/wiki/Preconditioner) when possible.
* Choose your initialization points wisely. For instance, if you are running many similar optimizations, warm-restart one with the results of another.
* Relax the tolerance if you don’t need precision using the parameter tol.

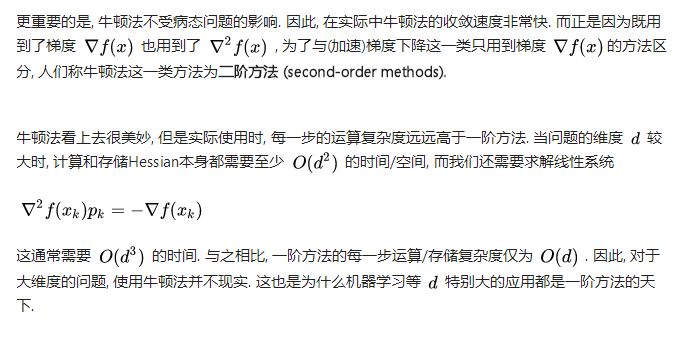
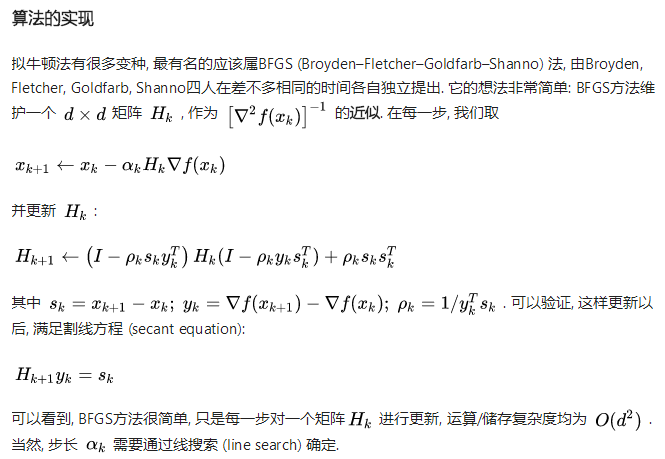
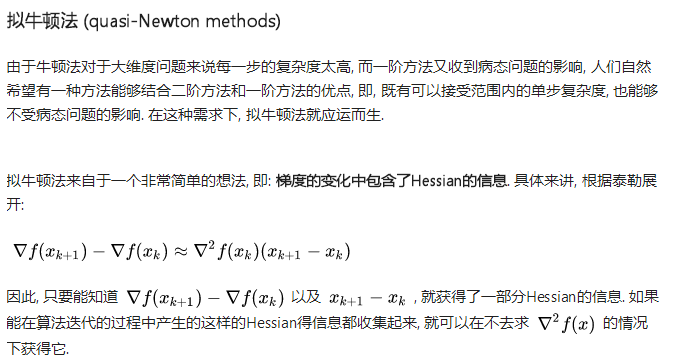
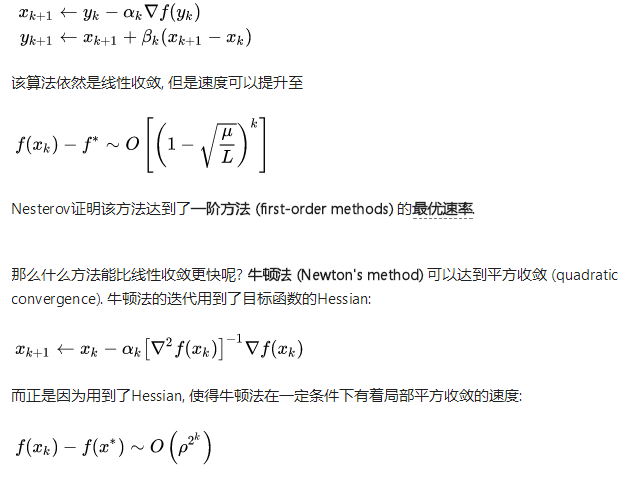
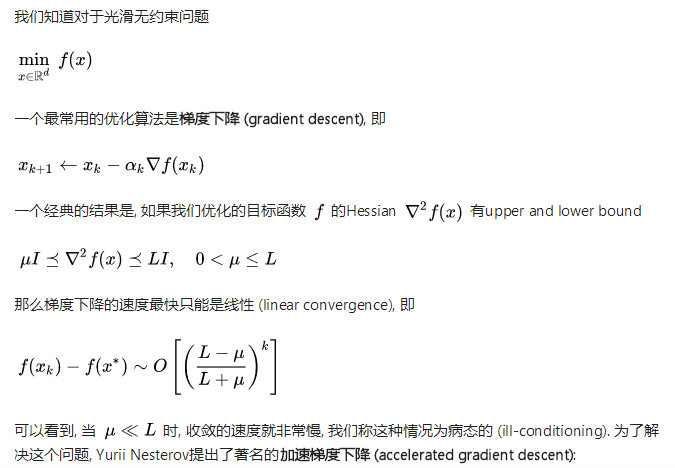
Python抽象基类(ABC)的意思，是定义一个虚拟的类， 不能直接调用方法， 只是一个接口， 相当于一个模板， 供他的子类使用，子类一定要实现他的方法。否则就会抛异常。  
疑问： 已经有了鸭子类型 和多态 ，为什么还要用这个呢？  
答： 为了 解决两个事情。

If the alpha factors are almost sure to decay after publication, can we take advantage of it?



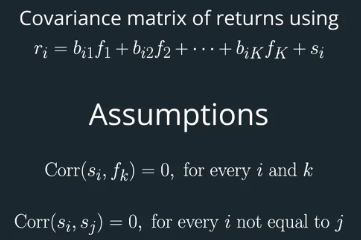


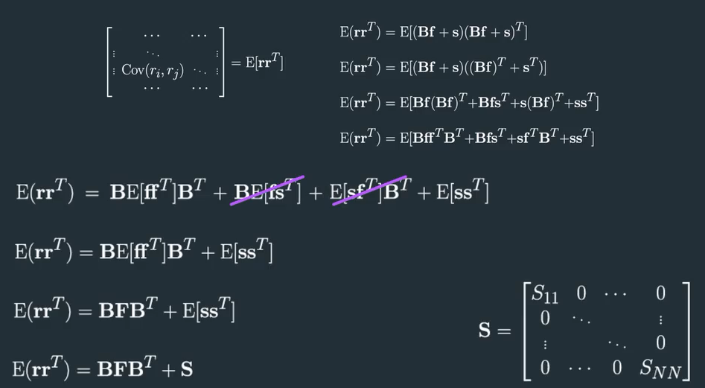


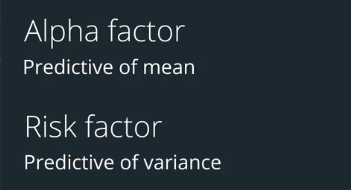


BFGS optimization is quasi-Newton method.

Some factors are hard to measure by appropriate data. Then, try to find and create these data.



How to represent covariance matrix by factor model.



Using alpha factors to predict the mean, and risk factors to predict the variance.

Both the mean and variance are important part to predict.

#### Risk Factors v. Alpha Factors

In general, risk factors are significant contributors to the variance of asset returns, and less predictive of the mean of returns. Risk factors are identified to control risk. One way to do control an asset's exposure to a risk factor is to hold an equal amount long as short. For instance, a dollar neutral portfolio with equal amounts long and short is controlling for risks that the overall market may move up or down.

In general, factors that are significant in describing the mean of asset returns can be candidates for alpha factors. Alpha factors are used to give some indication of whether each stock in the portfolio may have positive expected returns or negative expected returns. For example, a former alpha factor was the market capitalization of a stock. Small cap stocks tend to have higher future returns compared to large cap stocks.

What if a factor with both significant mean and variance??



Cash flow may directly measure the state of company’s finances. But it’s more volatile than earnings. While earnings can be smoothed and manipulated.

Some companies like Amazon may reduce their profit on purpose to avoid tax. So, even negative profit is not necessarily a downside indicator.

Orbital insight tracks the parking lots of JC Penny to tell its downward from 2012 to 2017.

Track Energy Information Administration(EIA) to estimate crude oil state.

Can we start with the return rankings of all companies and get a time series? Then find some thing in common with the stocks exhibit abnormal returns.

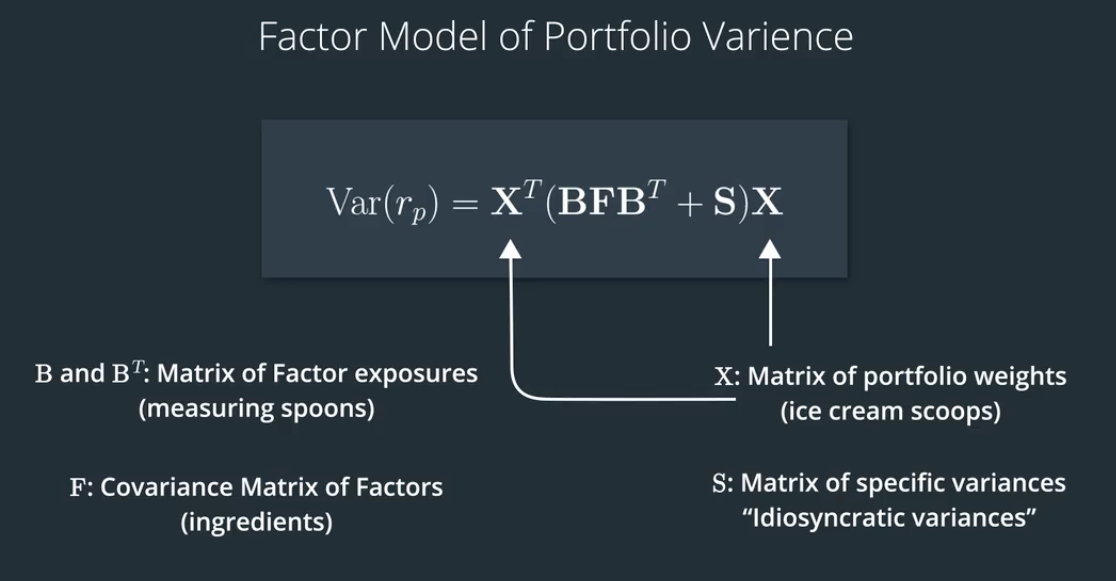
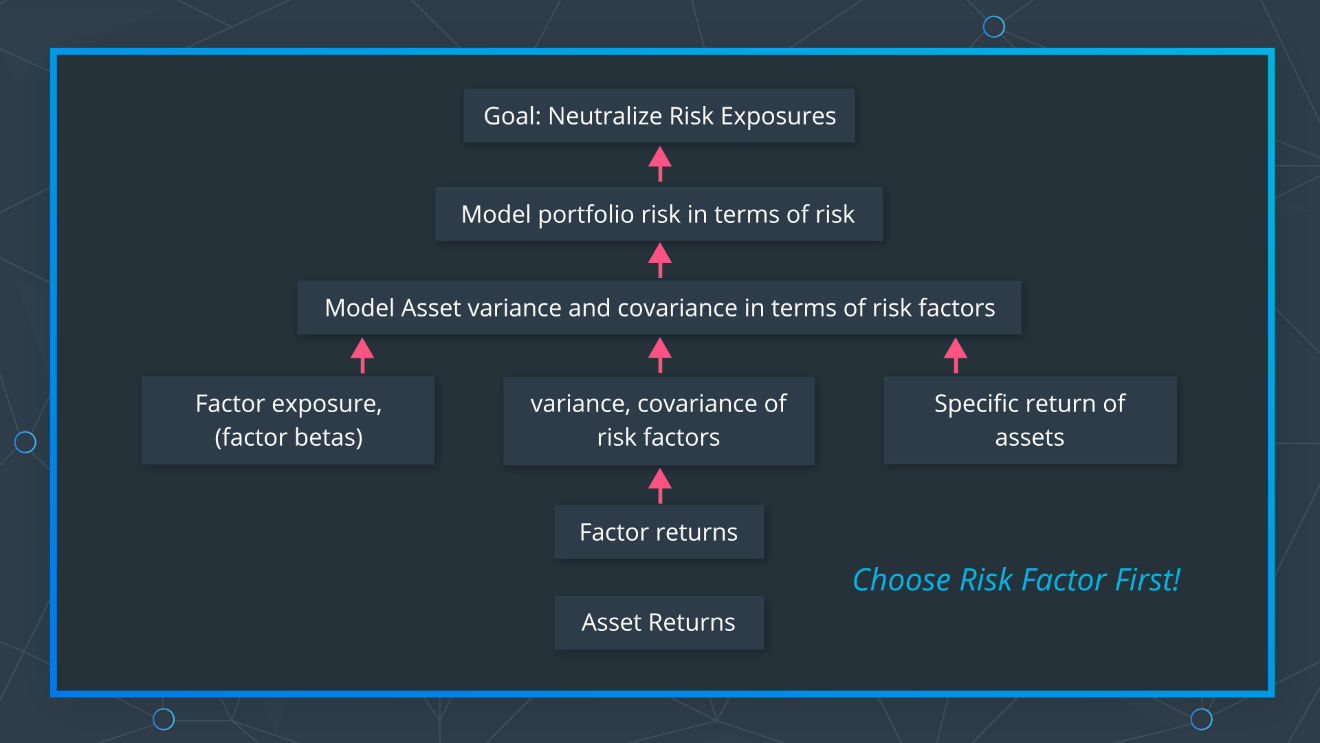
#### Some companies that are aggregating alternate data

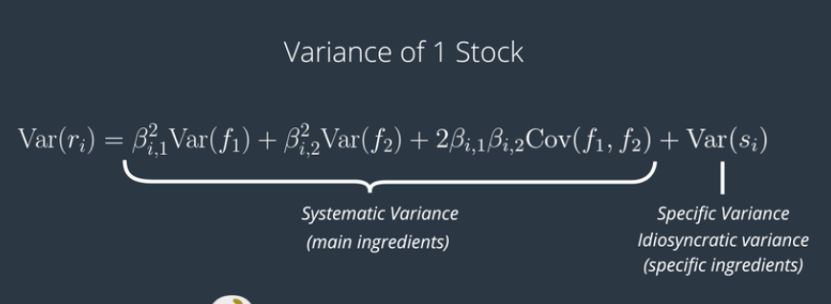
<https://www.buildfax.com/>

<http://edi.om/>

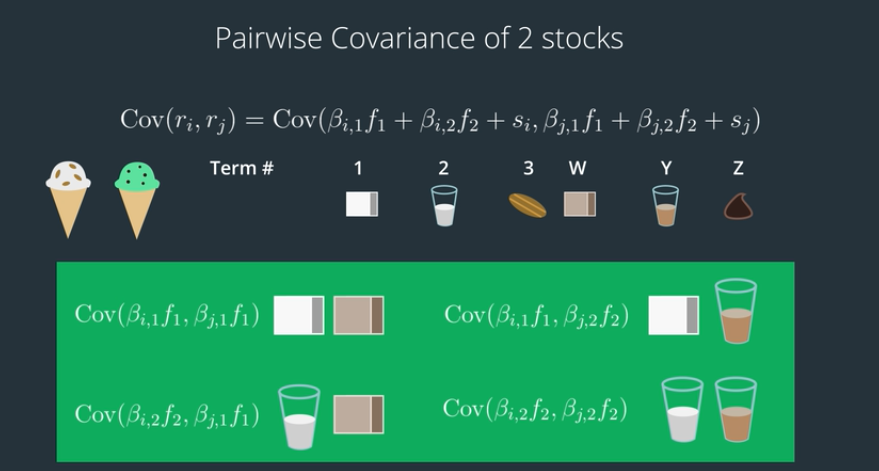
<https://www.thinknum.com/>

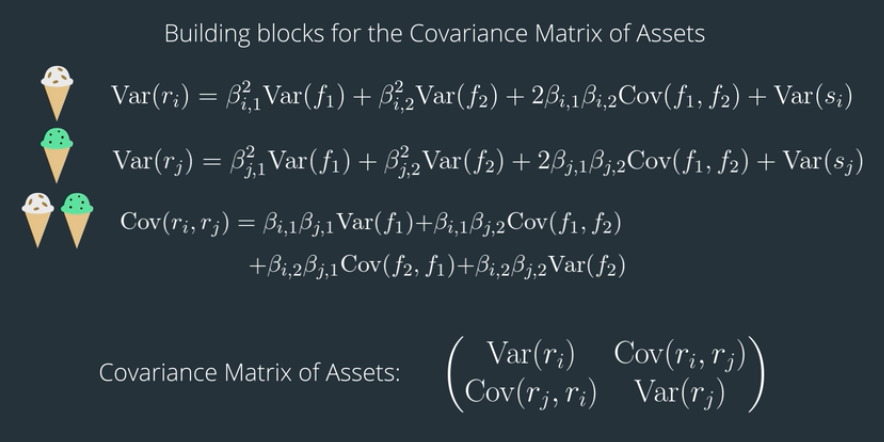
<https://orbitalinsight.com/>

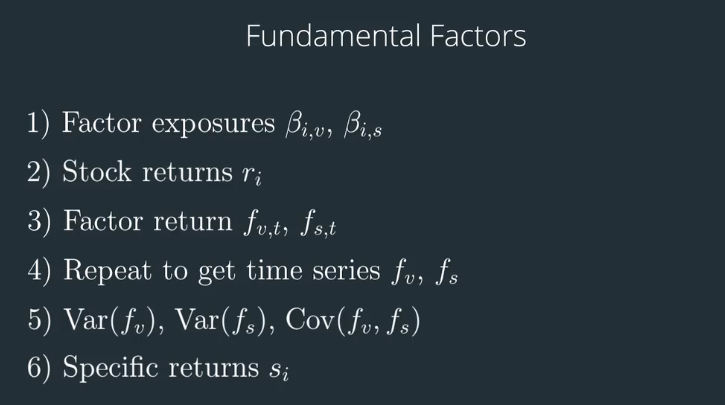
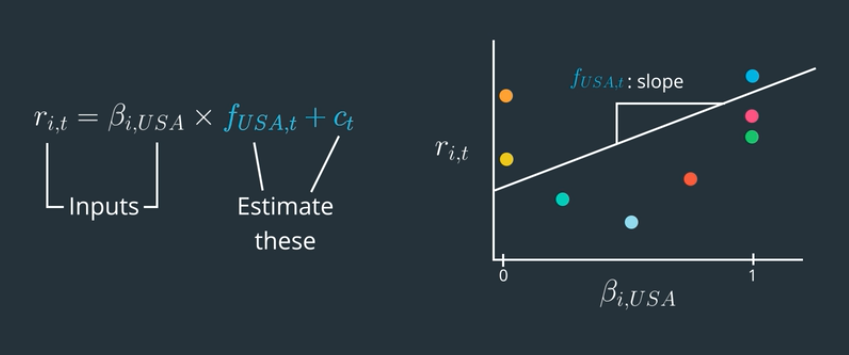
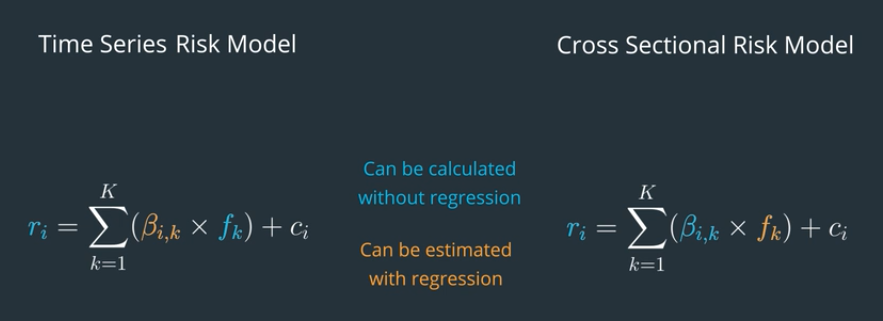




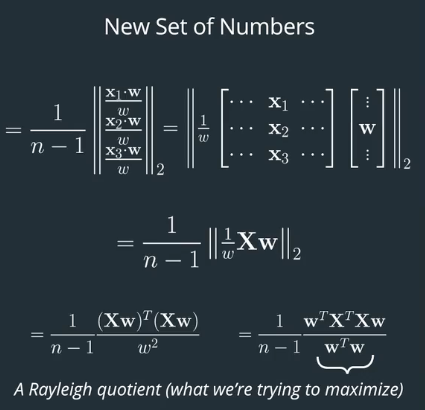
Factors with lower correlation will decrease the third term.







During the PCA process, the last basis will not maximize the variance along the basis, whereas all other basis do.



For PCA, the total variance is the same, if we keep the same number of basis and have them orthogonal.

