**Plotting the Yield Curve**

Data: U.S. Department of the Treasury

Got the data. I used the CubicSpline method in Python to create polynomials for each interval (2 consecutive observed data points). The method solves the equations for you and derives the parameters of the polynomials. I need this method because if you plot the data points alone, you will not get a smooth curve; it is more like a curve connecting points. I evaluate the yield for maturities not given by the U.S. Department of the Treasury by modeling those intervals as polynomials and using the method np.linspace to generate equally distanced points (yield, maturity) following the polynomial of each interval. Doing this results in the yield curve.

So now that I know how the yield curve looks like and what its evolution has been leading up to the latest date I can start thinking about ways of predicting it. One way of doing is is by adopting the Principal Component Analysis (PCA) method using Singular Value Decomposition (SVD). PCA helps to reduce the dimensionality of the data while capturing the most significant patterns. When I say dimensionality I mean the number of features/columns.variables that we have data on. PCA will tell which features are most valuable for clustering the data. To give you an intuition of what the PCA does suppose we only have 2 dates in our data frame. We want to plot the data points on a 2D X/Y graph. Say that day 1 is the x-axis (spans one dimension of the graph) and day 2 is the y-axis (spans the other dimension). See this pic:

A graph with blue dots and red lines

Description automatically generated

If we were to cluster these points, then one could separate them by the left and right of the vertical line at 4.75. This is a 2D plot, but imagine adding another day, and now you deal with a 3D plot and so on. That is what we refer to by dimensionality, and that is how it impacts what we do. 3D is the most we can capture by a visualization, so PCA is quite useful when working with more than 3 dimensions because it helps reduce those dimensions while limiting the errors such that we can visualize the data in fewer dimensions. It helps with deciding which dimensions/days we should not overlook. As perhaps guessed, the PCA also determines which date is the most valueable for lustering data points.

If I calculate the averages across dates I know by how much I can shift the points by so that that center (average1,average2) becomes the origin. I center the points.

A graph with blue dots and red lines

Description automatically generated

Then, you have to fit a line through the points, and there are several ways to look at it. The PCA will take the projection of the points on a random line that is first drawn (intersecting the origin). It then wants to minimize the sum of the squared distance of the projections to the origin and rotates the line to achieve that. As an aside, the eigenvalue is the average of the sum of squared distances. This is what the PCA will do, but I assume a linear regression will result in the same line fit. The fit line is called the Principal Component 1 (PC1). You can then get the slope of PC1.

Now, you want to get the unit vector/eigenvector of PC1, which is done by dividing by the length of the vector given by the slope. This is to standardize values, nothing else because the slope remains the same. The reason for this standardization is to derive meaningful info because numbers between 0 and 1 can represent percentages, but without standardization, it’s hard to derive any info. The slope tells you the linear combination of the 2 axes. Standardizing tells you how much of each axis contributes to the dispersion of the points.

PC2 is the perpendicular to the PC1 at the origin. This gives you the slope of PC2, and you derive the eigenvector of PC2. Then, projecting the points onto PC2, you will get the eigenvalue for PC2.

By rotating the PCs so that they become the axes of a graph and knowing the coordinates of the points, you replot the points. This time around, they are standardized, so you will clearly see the variation on each axis. This step is not necessary, but it helps you visualize the variation more easily.

Eigenvalues are measures of variation. You know how much the date1 accounts for total variation (total variation = sum of eigenvalues across PCs). So, the eigenvalue of PC1 / Sum eigenvalues gives you a percentage of the total dispersion of PC1. The goal is to select the PCs with the highest contribution (%) to total variation. So, in a 3D if 2 PCs account for 99% of the variation then plotting a 2D using those 2 dates represented by those 2 PCs is a pretty good representation of the variation in the data points. Imagine you got to the point of 3PCs. PC3 is perpendicular to PC1 as PC2 since you have 3 dimensions. So, if PC3 does not account for much variation, then erase it, project the points on the remaining PCs, and carry on as if you were working with only 2 PCs.

So again, the PCA tells us what we must focus on to account for the variation of data points as best as possible, given a set constraint on the dimensions you can use. Say at most 2 or 3. We try to simplify a problem while not creating large errors. At the end of the day, it is a tradeoff between simplicity and accuracy, and you want to hit the sweet spot for your constraint.

As you might know, the eigenvalue tells you the magnitude, and the eigenvector tells you the direction of the divergence.

This is a scree plot showing the relevance o the PCs:

The dimensionality of my data is given by the number of variables/columns (i.e., dimensionality = 13). So, I want to reduce the number of maturities I will consider while preserving as much of the data’s variance as possible. Why do it? Well, the sole meaningful benefit is to reduce the noise so I can focus on the most significant patterns. For example, how correlated is the 1 month with the 2 months compared to the 1 month with the 10 y. Why take in both the 1 month and the 2 months if the actual significant pattern can be given by one of them?

I felt the need to explain intuitively what is happening under the hood because the code is short, and the objects and methods were already built in, but having somewhat of an understanding of the inner workings might help.

If PC1 and PC2 explain most of the variance, focusing on features with high loadings on these PCs can help you understand which original features are most influential.

So, the loadings are the coefficients in the linear combination, giving a PC. The sum of the squared loadings for a PC is equal to 1. So the size of the loadings can be thought of as a weight of the factor for the PC.

So once I chose my PCs, I got the loads of all the factors for each PC. Square them now they’re weights. Time the respective weight by the respective PC value (%) and sum up for all PCs to get the overall impact of a factor on the data frame’s variance. Now, choose which factors/variables/columns (here maturities) to eliminate.

Use that first sheet of the xlxs from statista to exl=plain the indicators like what they stand for.

I think like understanding what can be used in the Lsso as predictor variable and what can not is important. Like can you have the 10y if u wanna predict the cure? Probably not. So this applies to all the yields and technically the FF rate and spreads containing them.