1. **Pipeline in ML**: **check Jupyter notebook, please(for code)...**

**Jupyter Notebooks:**

* Normalization and Standardization
* Pipelines-Using-Sklearn
* Pipeline\_Scikit\_Learn

Sequentially apply a list of transforms and a final estimator. **Intermediate steps of the pipeline must implement fit and transform methods and the final estimator only needs to implement fit.**

As the name suggests, ***pipeline class allows stacking multiple processes into a single scikit-learn estimator. pipeline class has fit, predict and score method just like any other estimator* (ex. LinearRegression).**

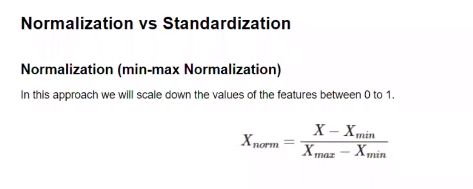
* The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters.

1. **MinMaxScaler: check Jupyter notebook, please(for code)...**

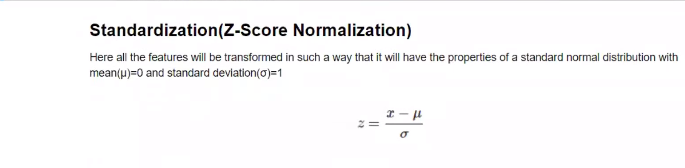
**Note:**

1. **Normalization** will help you to scale down your features between 0 and 1. Eg: **MinMaxScaler**
2. **Standardization** will help you to scale down your features based on the standard normal distribution (mean is 0 and sd is 1). Eg: **StandardScaler**

**MinMaxScalar:**

****

**3) StandarScalaer: check Jupyter notebook, please(for code)...**

****

4) **PCA(Principal Component Analysis):**

**from sklearn.decomposition import PCA**

**pca = PCA(n\_components = 2)**

**X\_train = pca.fit\_transform(X\_train)**

**X\_test = pca.transform(X\_test)**

# **What is PCA? (**[**https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c**](https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c)**)**

* Let’s say that you want to predict what the [gross domestic product](http://www.investopedia.com/terms/g/gdp.asp) (GDP) of the United States will be for 2017. You have lots of information available: the U.S. GDP for the first quarter of 2017, the U.S. GDP for the entirety of 2016, 2015, and so on. You have any publicly-available economic indicator, like the unemployment rate, inflation rate, and so on.
* You have U.S. Census data from 2010 estimating how many Americans work in each industry and [American Community Survey](https://www.census.gov/programs-surveys/acs/about.html) data updating those estimates in between each census.
* You know how many members of the House and Senate belong to each political party. You could gather stock price data, the number of [IPOs](https://en.wikipedia.org/wiki/Initial_public_offering) occurring in a year, and [how many CEOs](https://www.nytimes.com/2017/03/09/business/bloomberg-iger-business-executives-president.html?_r=0) [seem to be mounting a bid for public office](http://www.latimes.com/business/technology/la-fi-tn-zuckerberg-president-20170120-story.html). Despite being an overwhelming number of variables to consider, this *just scratches the surface*.
* If you’ve worked with a lot of variables before, you know this can present problems. Do you understand the relationships between each variable? Do you have so many variables that you are in danger of overfitting your model to your data or that you might be violating assumptions of whichever modeling tactic you’re using?
* You might ask the question, “How do I take all of the variables I’ve collected and focus on only a few of them?” In technical terms, you want to “reduce the dimension of your feature space.”
* By reducing the dimension of your feature space, you have fewer relationships between variables to consider and you are less likely to overfit your model. (Note: This doesn’t immediately mean that overfitting, etc. are no longer concerns — but we’re moving in the right direction!)
* Somewhat unsurprisingly, *reducing* the *dimension* of the feature space is called “*dimensionality reduction*.”

# **When should I use PCA?**

1. Do you want to reduce the number of variables, but aren’t able to identify variables to completely remove from consideration?
2. Do you want to ensure your variables are independent of one another?
3. Are you comfortable making your independent variables less interpretable?