Mathematical variable transformation:

Variables should be there for linear reg

Histogram and q-q plots

More oftenly- not normal distribution

To bring- mathematical transformation

* Variable trasformations:
* Log- log(X)
* Exponential: X exe(lamda) or sqr(X)/Cube(X)
* Reciprocal: 1/X
* Exponential:
* Box-cox:
* Yeo-johnson

**Gaussian Transformation**

Some machine learning models like **linear and logistic regression** assume that the variables are normally distributed. Often, variables are not normally distributed, but, transforming the variables to map their distribution to a Gaussian distribution may, and often does, boost the performance of the machine learning algorithm.

If a variable is not normally distributed, it is often possible to find a **mathematical transformation to normalise its distribution.**

**How can we transform variables so that they follow a normal distribution?**

The most commonly used methods to transform variables are:

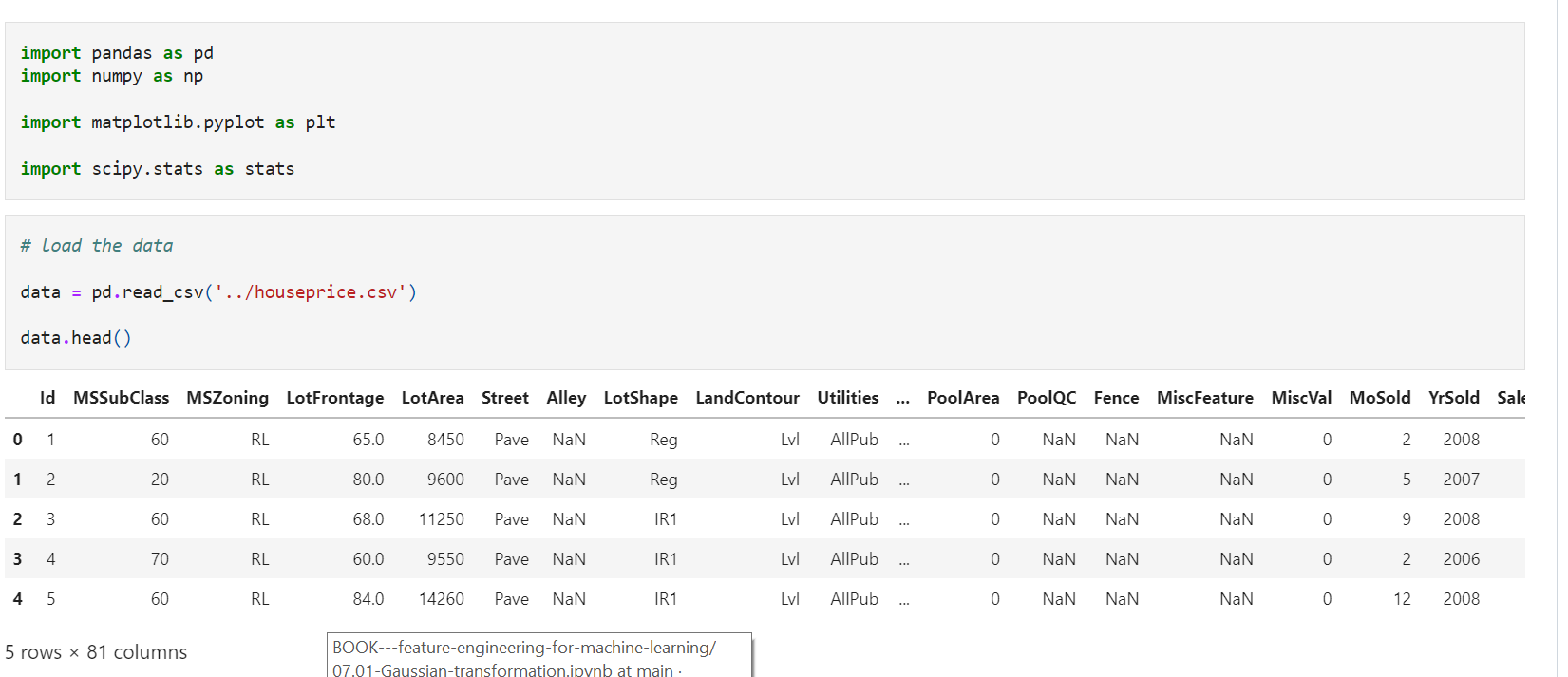
* Logarithmic transformation - np.log(X)
* Reciprocal transformation - 1 / X
* Square root transformation - X\*\*(1/2)
* Exponential transformation (more general, you can use any exponent)
* Box-Cox transformation
* Yeo-Johnson transformation

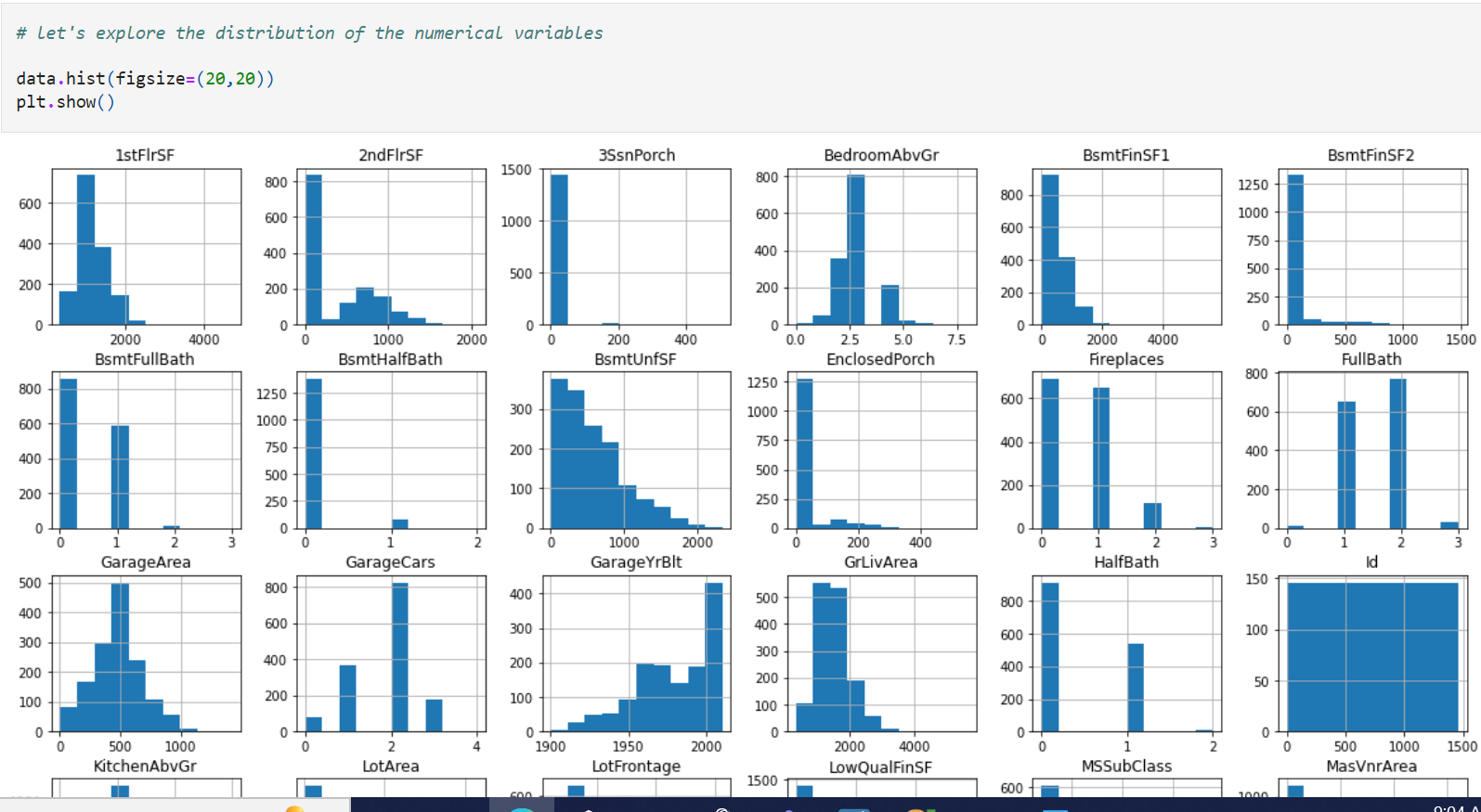
You can find the formulas for Box-Cox and Yeo-Johnson [here](https://scikit-learn.org/stable/modules/preprocessing.html#mapping-to-a-gaussian-distribution)

Briefly, the **Box-Cox transformation** is an **adaptation of the exponential transformation**, **scanning through various exponents**, and it already represents the untransformed variable, as well as **the log transformed, reciprocal, square and cube root transformed**, as the **lambda varies across the range of -5 to 5 (**see formula or accompanying video, to understand this better).

So by doing **Box-Cox transformation, in a way**, we are **evaluating all the other transformations and choosing the best on**e. Box-Cox can only be applied to **positive variables.**

Yeo-Johnson is **a modification of the Box-Cox transformati**on so that it can be applied as well to **non-positive variables**



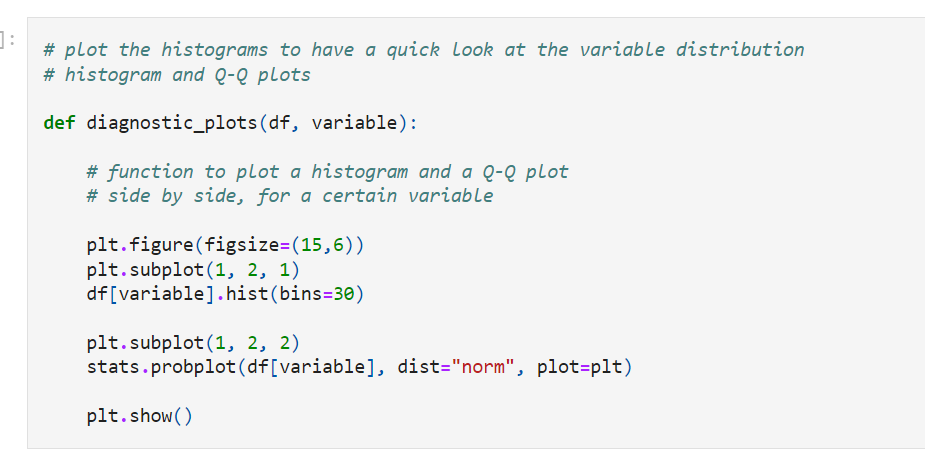


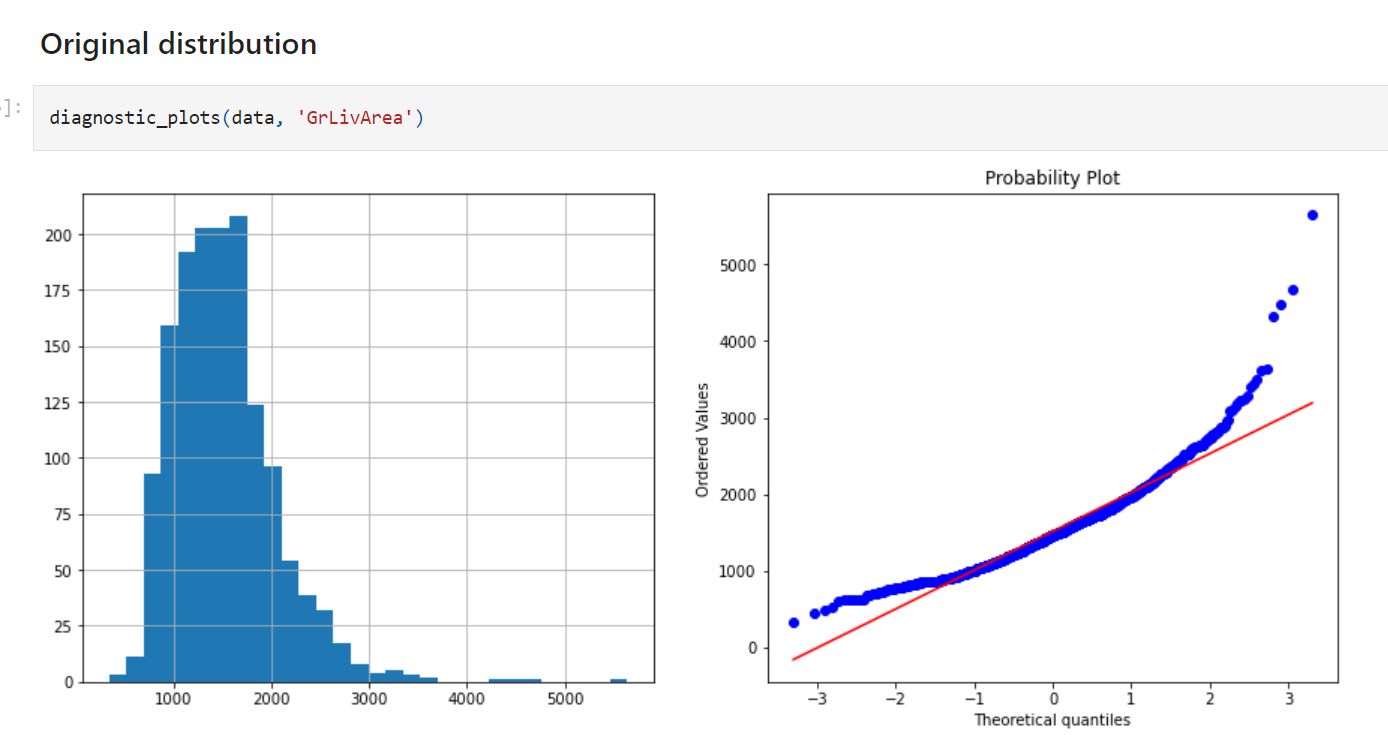
**Plots to assess normality**

To visualise the distribution of the variables, we plot a **histogram and a Q-Q plot**.

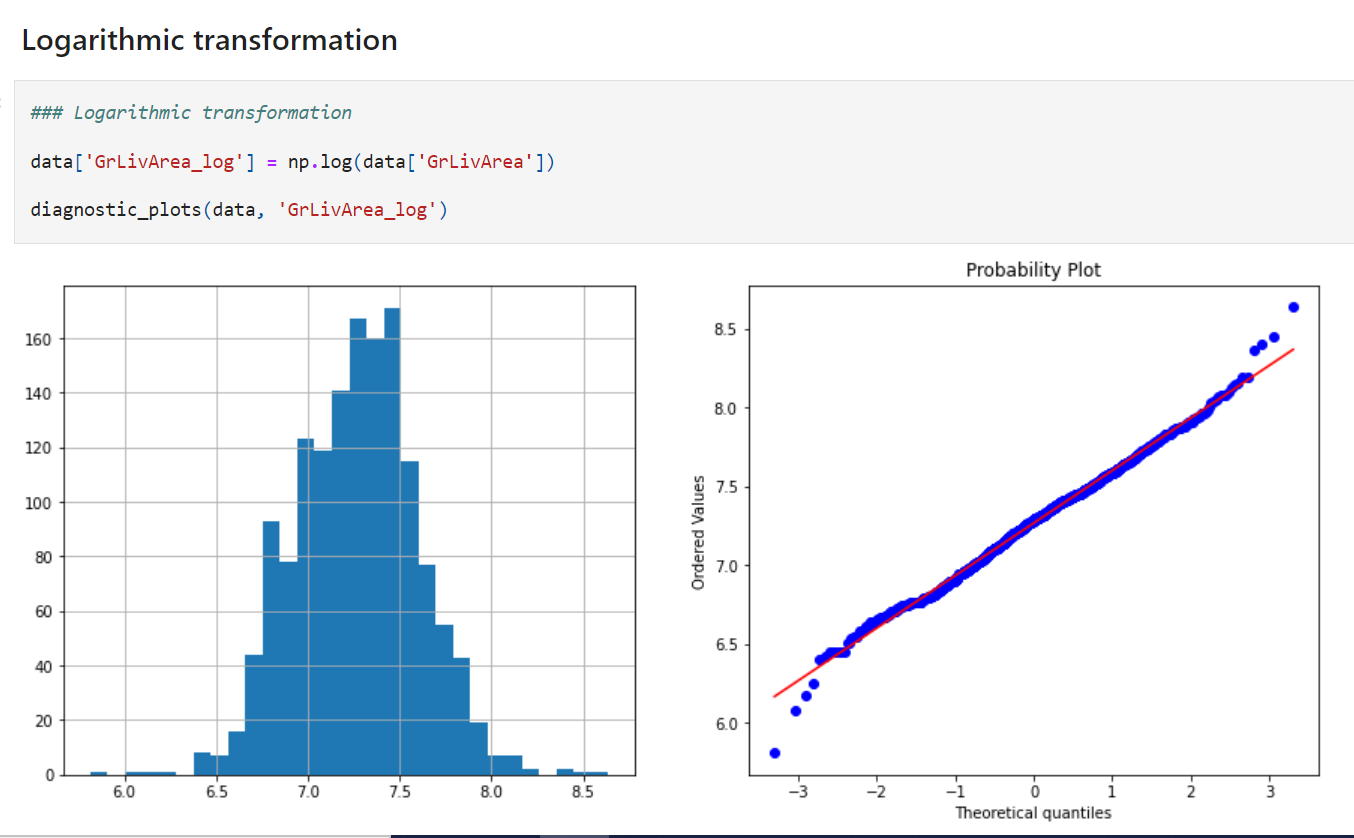
Q-Q pLots:

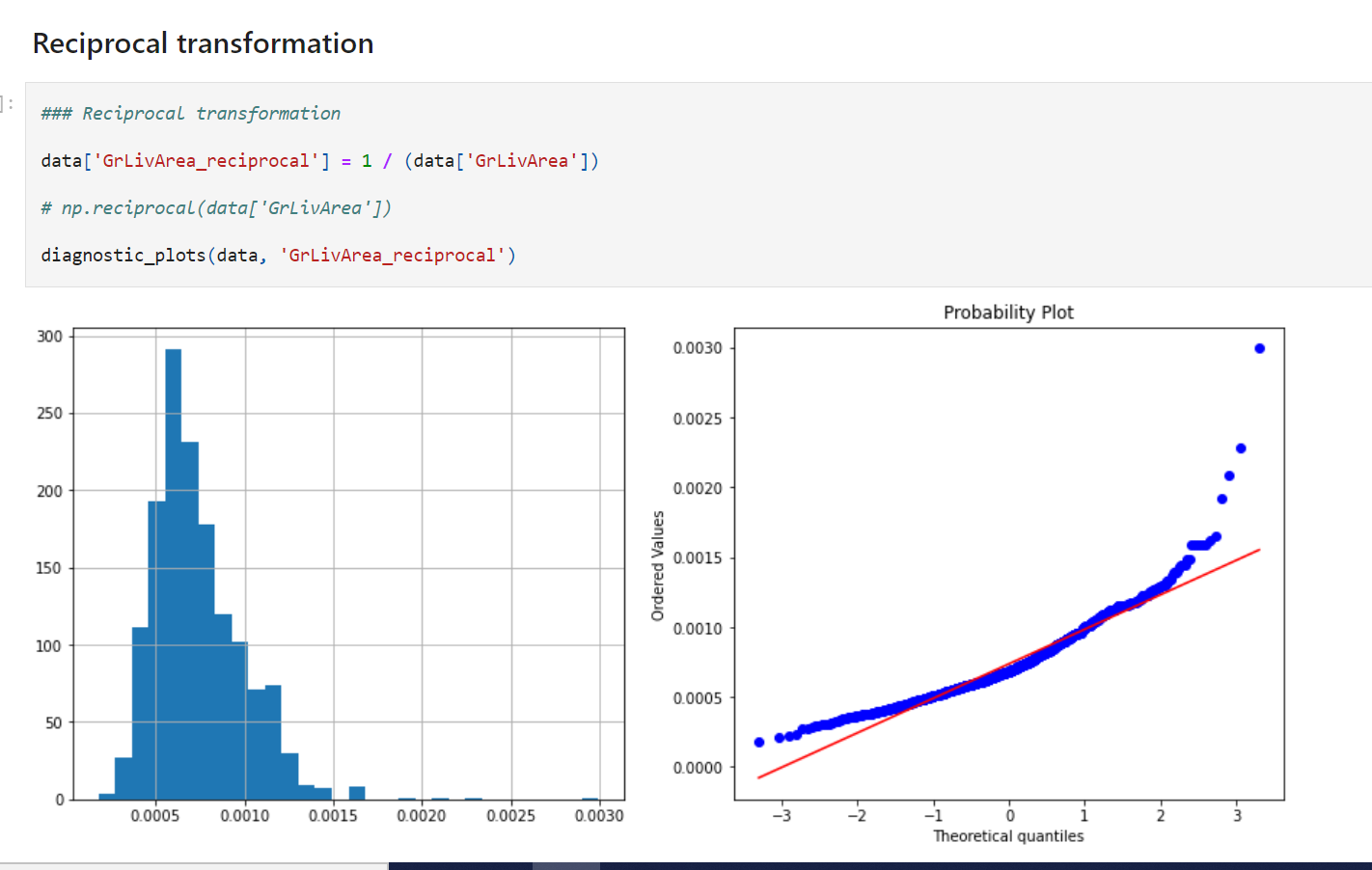
, if the variable is normally distributed, the values of the variable **should fall in a 45 degree line** when plotted against the theoretical quantiles. We discussed this extensively in Section 3 of this course.

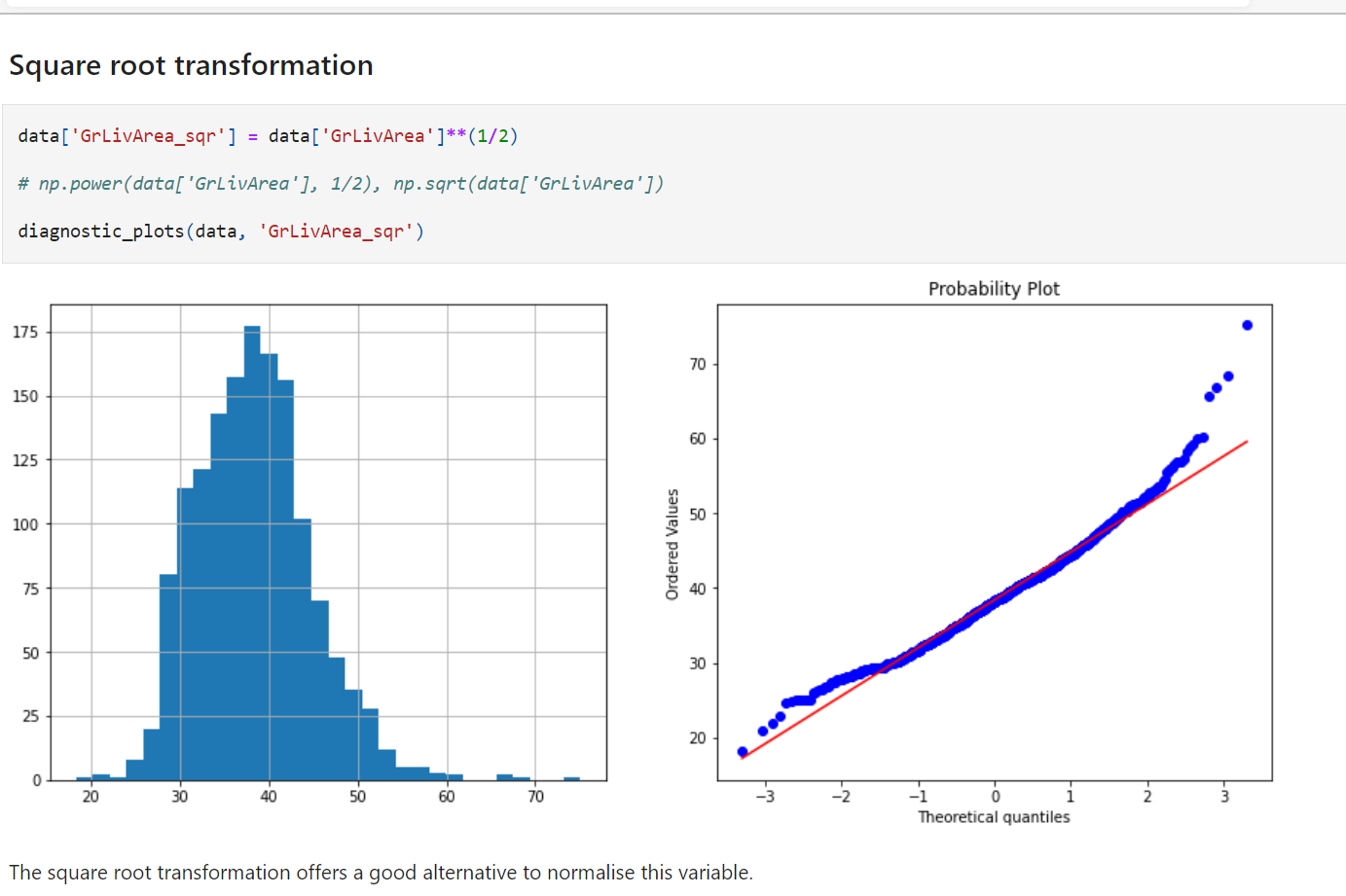




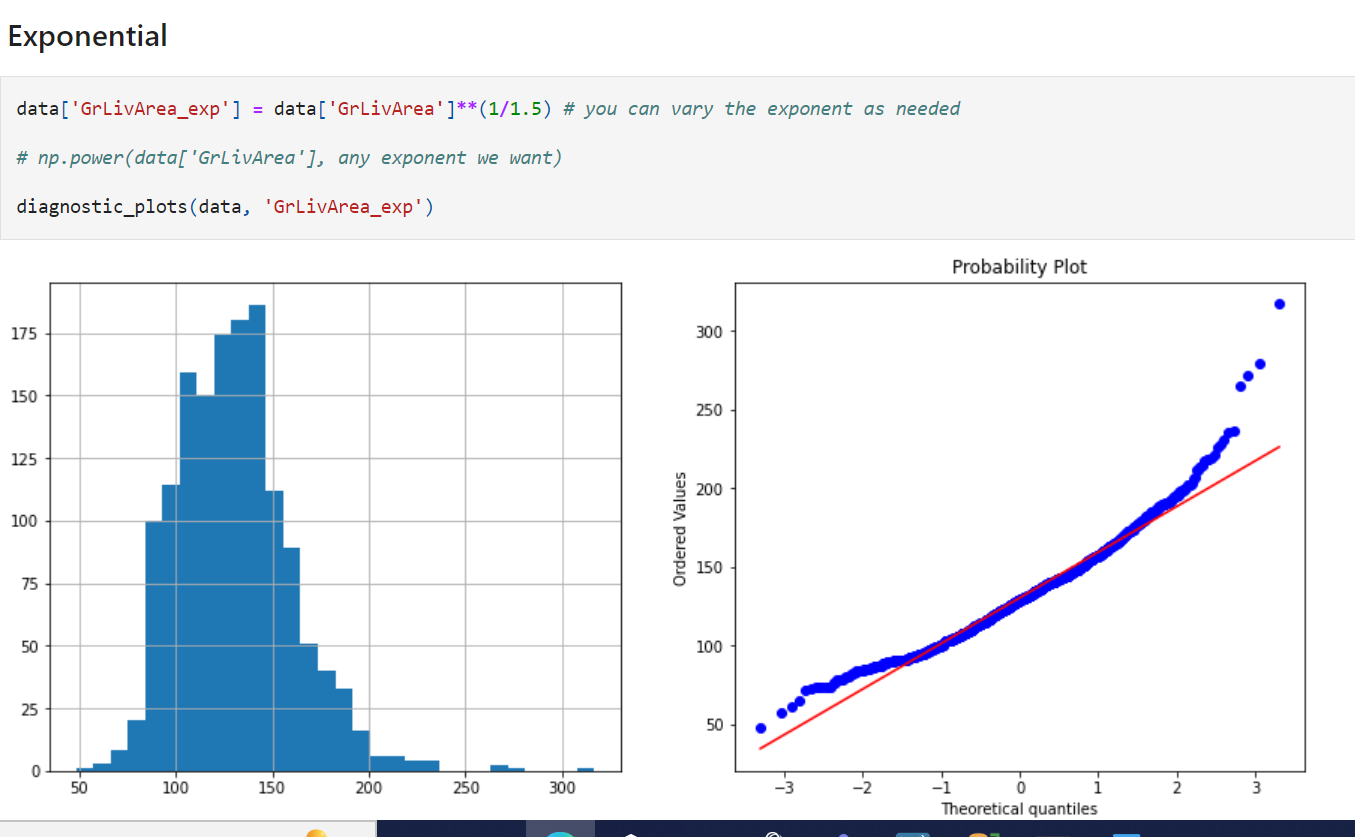
We can see in the plots that the **variable is not normally distributed**. The values **depart from the red line towards the ends of the distribution** and we can see in the histogram that it is skewed to the right.







The square root transformation offers a good alternative to normalise this variable.



The exponential transformation did not work so nicely for this variable.

**Box-Cox transformation**

The Box-Cox transformation is defined as:

T(Y)=(Y exp(λ)−1)/λ if λ!=0, or log(Y) otherwise.

where **Y is the response variable**

**λ is the transformation parameter**.

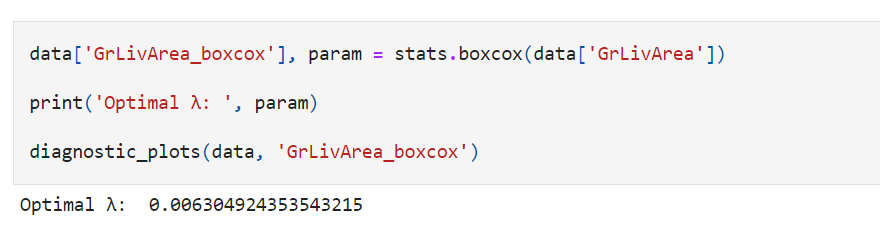
**λ varies from -5 to 5.** In the transformation, **all values of λ are considered** and the **optimal value** for a given variable is selected.

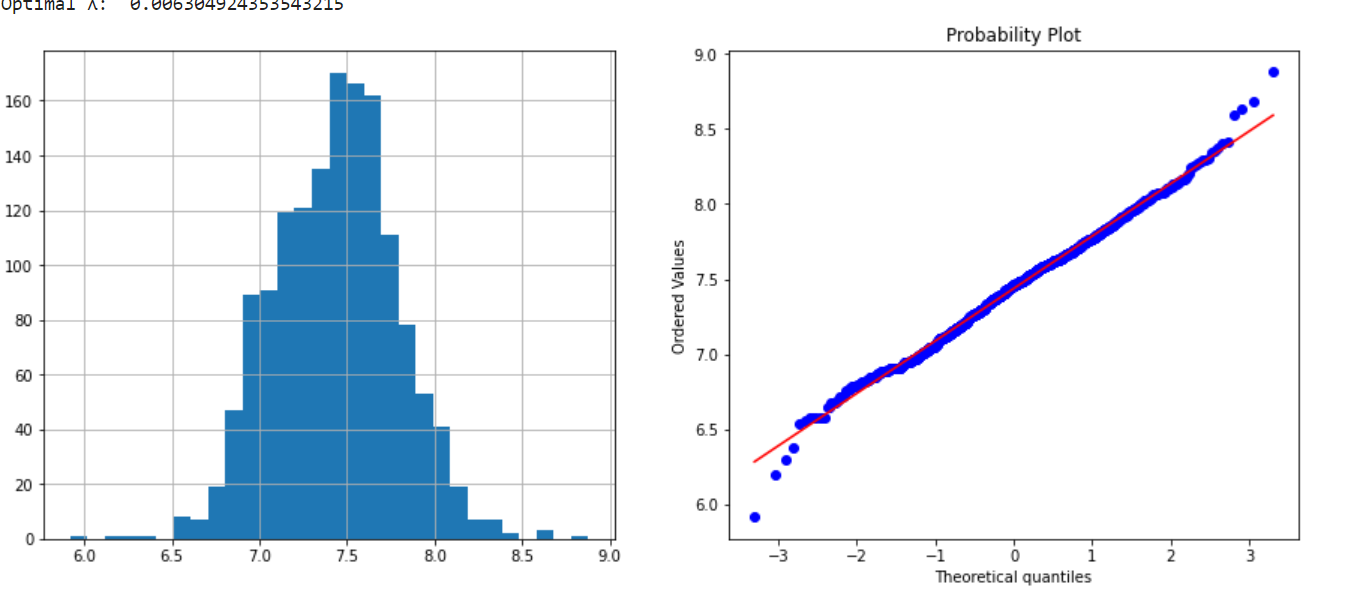
Briefly, for each λ (the transformation tests several λs), the correlation coefficient of the Probability Plot (Q-Q plot below, **correlation between ordered values and theoretical quantiles**) is calculated (this optimisation equation actually varies with the implementation).

**The value of λ corresponding to the maximum correlation** on the plot is then the optimal choice for λ.

In python, we can evaluate and obtain the best λ with **the stats.boxcox** function from the package scipy.

Let's have a look.

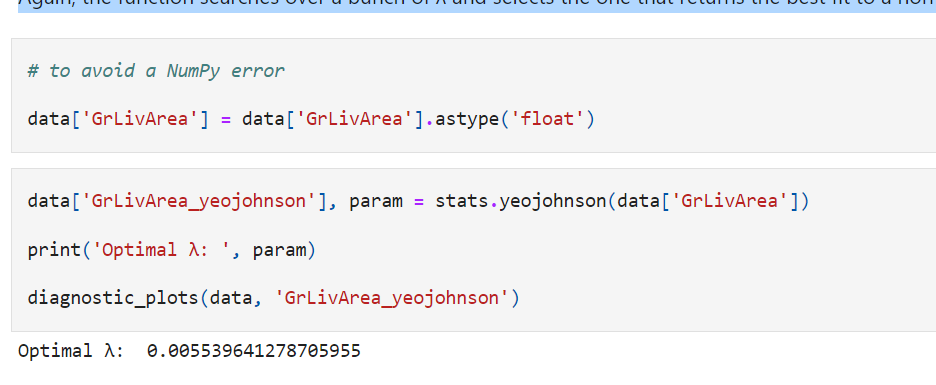


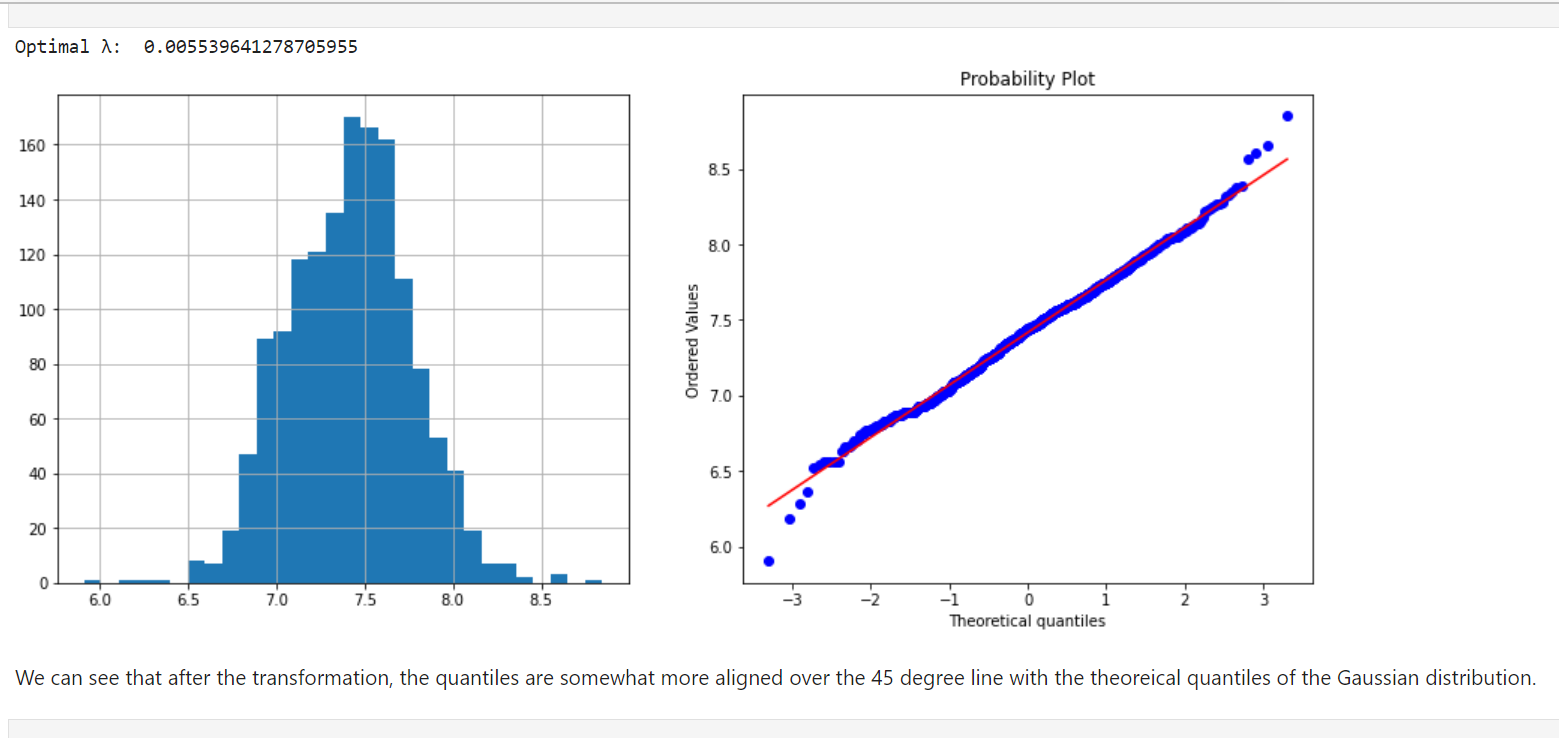


**Yeo-Johnson**

Yeo-Johnson is the **same as Box-Cox for the positive values** of the variable, but it has different **equations for the negative values** of the variable as described [here](https://www.stat.umn.edu/arc/yjpower.pdf)

Again, the function **searches over a bunch of λ and selects the one that returns** the best fit to a normal distribution





**Gaussian Transformation with Scikit-learn**

Scikit-learn has recently released **transformers to do Gaussian mappings** as they call the variable transformations

. The PowerTransformer allows to do **Box-Cox and Yeo-Johnson transformation**. With the FunctionTransformer, we can specify any function we want.

The transformers per se, do **not allow to select columns**, but we can do so using a **third transformer, the ColumnTransformer**

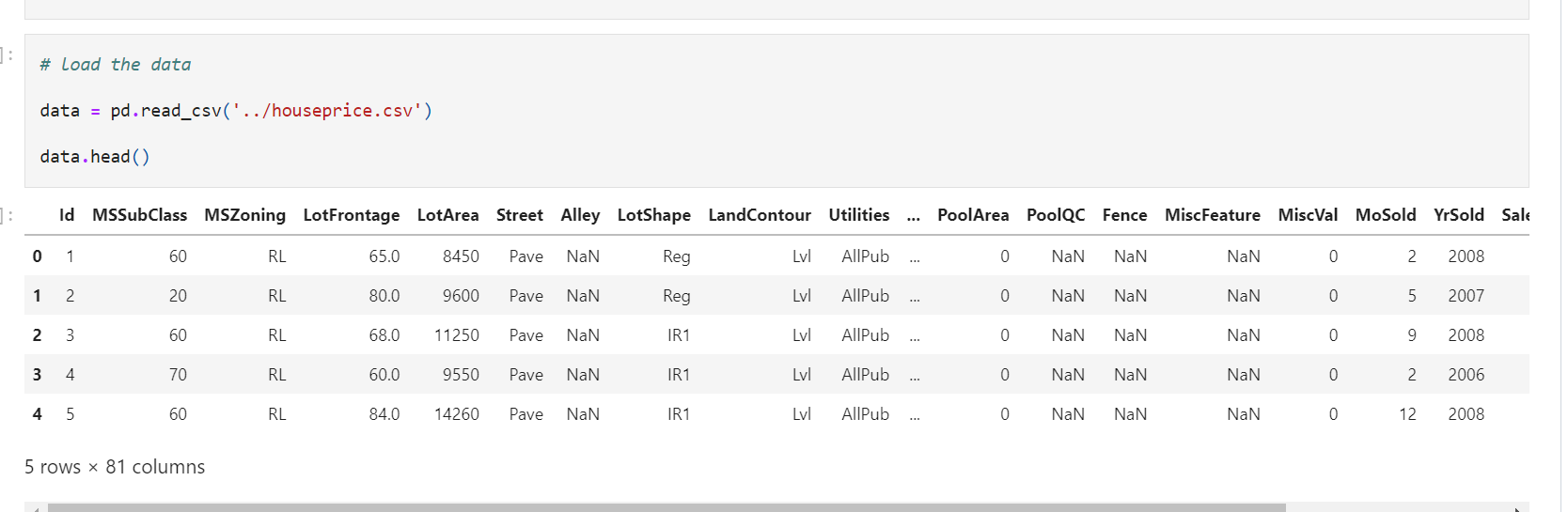
Another thing to keep in mind is that **Scikit-learn transformers return NumPy arrays,** and **not dataframes,** so we need to be mindful of the order of the columns not to mess up with our features.

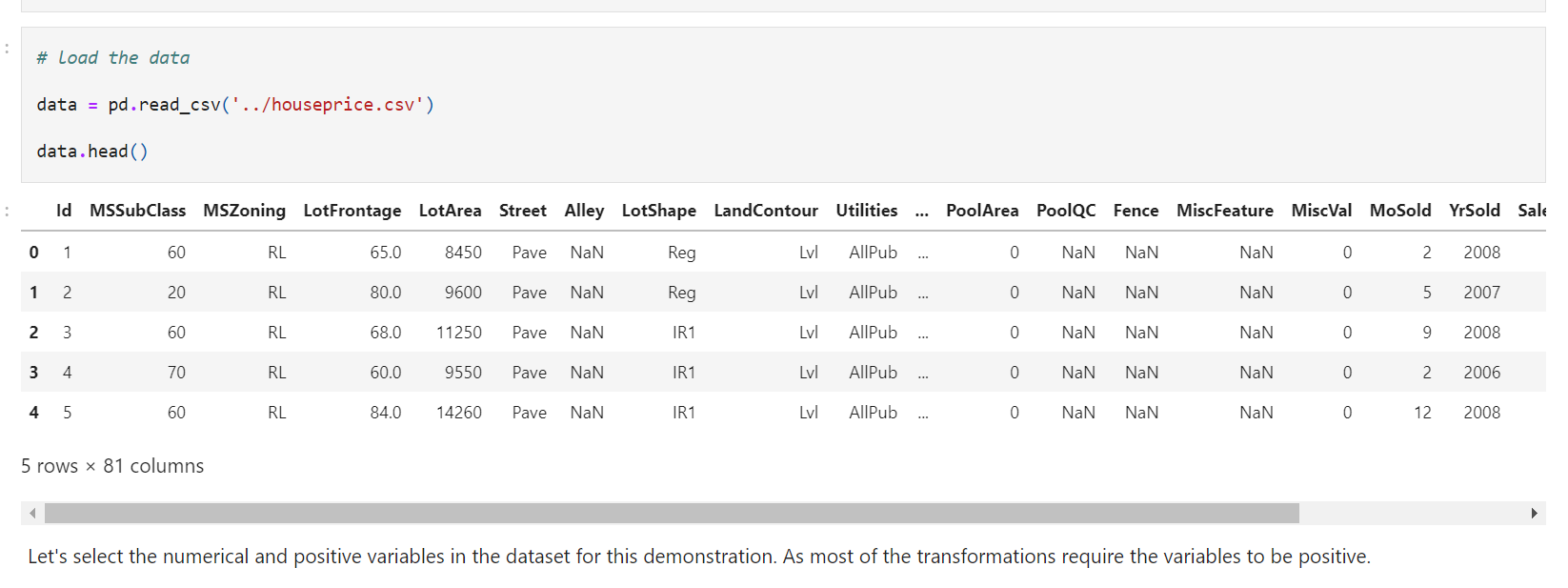
**Important**

Box-Cox and Yeo-Johnson transformations need to **learn their parameters from the data.** Therefore, as always, before attempting any transformation it is important to divide the dataset into **train and test set.**

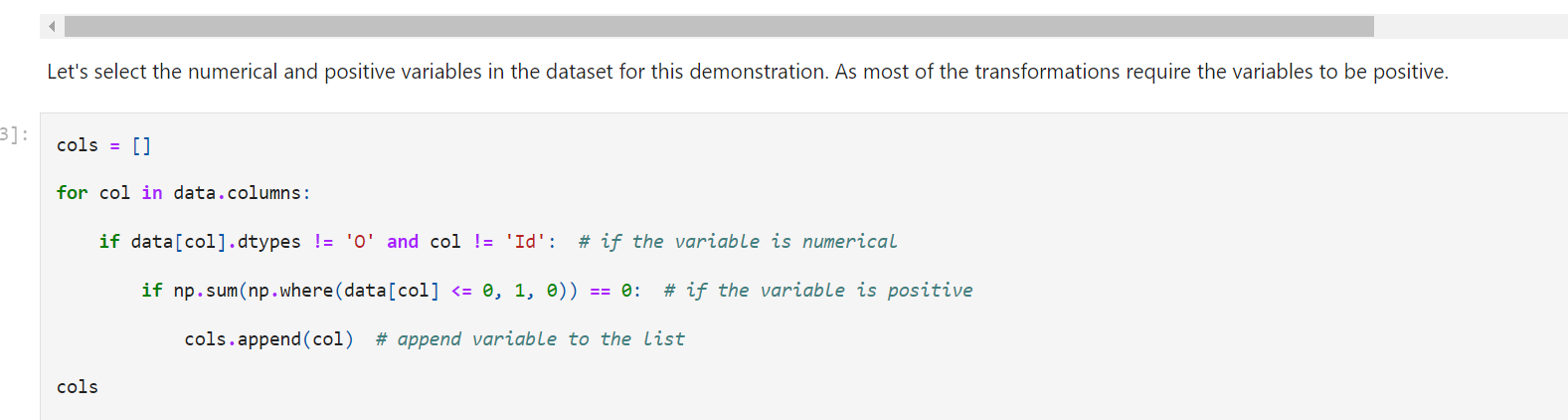
In this demo, I will not do so for simplicity, but when using this transformation in your pipelines, please make sure you do so.

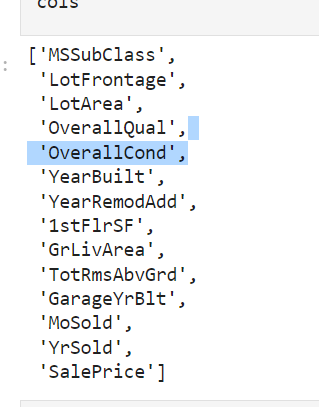
**In this demo**

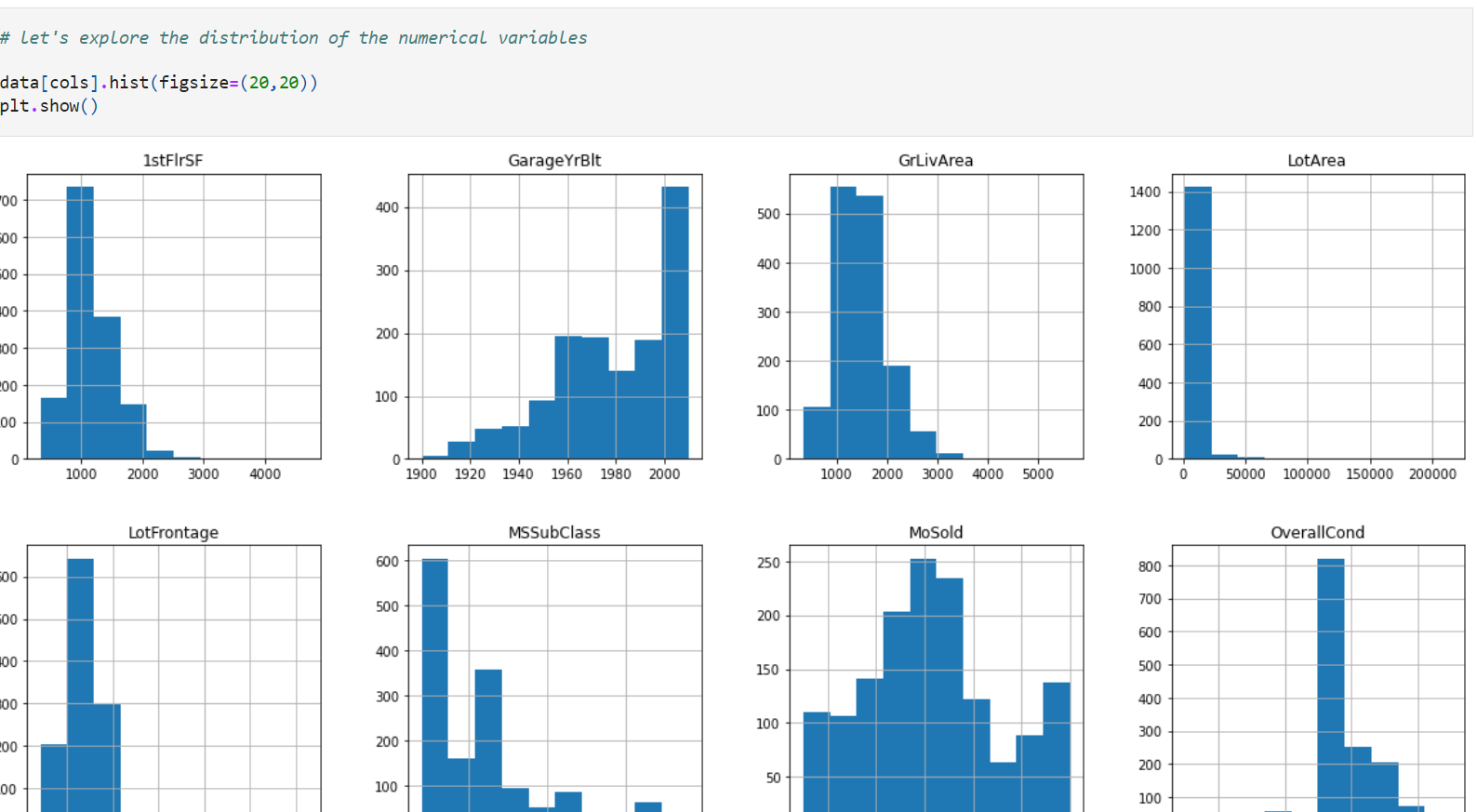




Let's select the numerical and positive variables in the dataset for this demonstration. As most of the transformations require the variables to be positive.



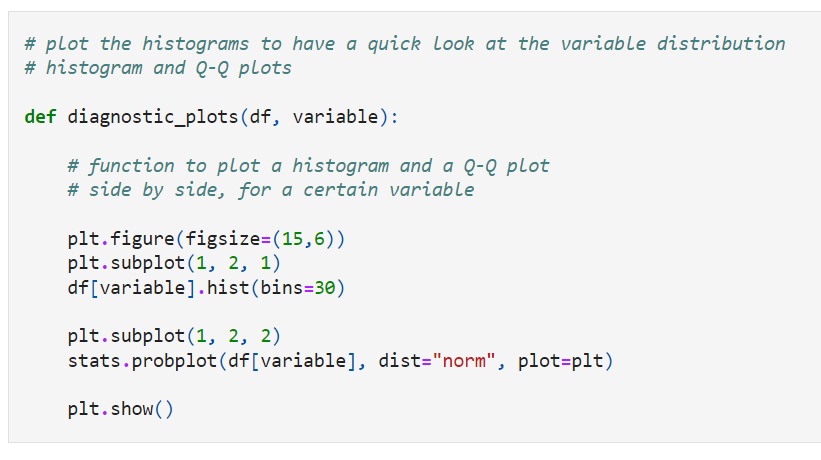


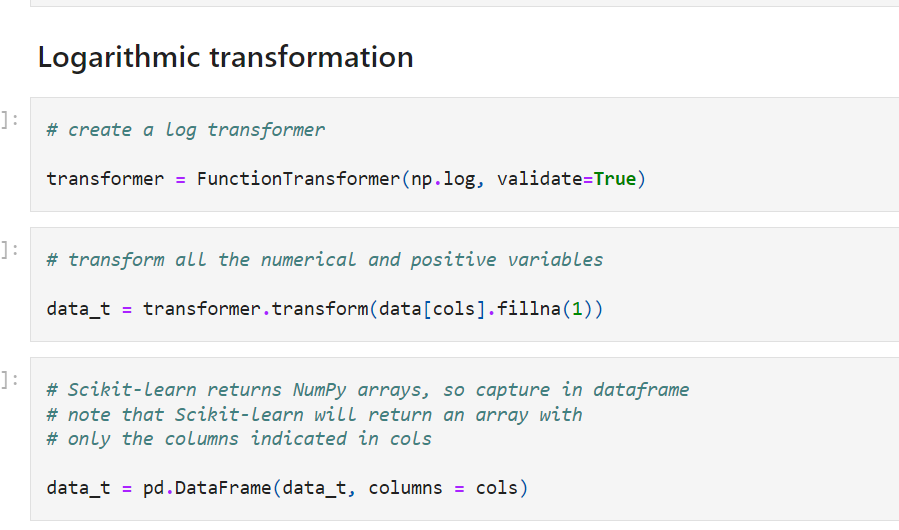


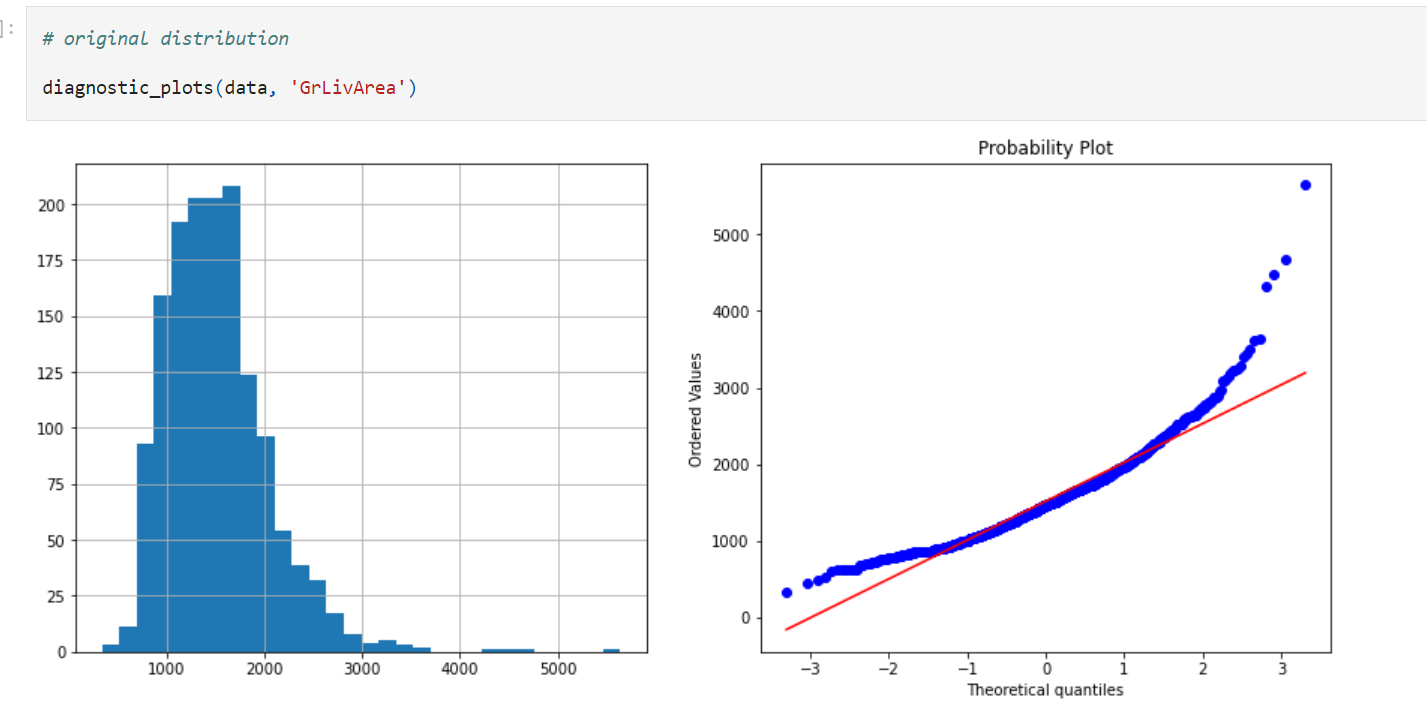
**Plots to assess normality**

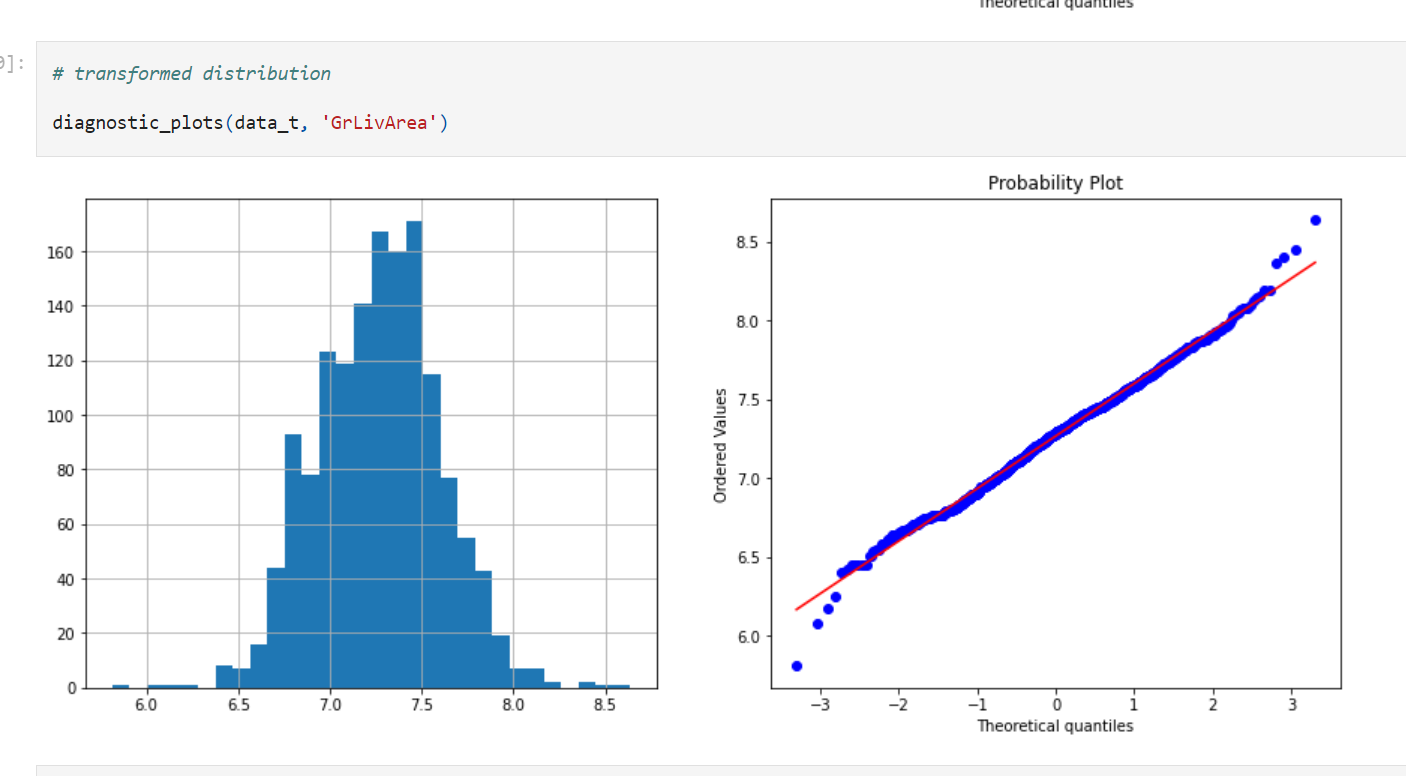
To visualise the distribution of the variables, we plot a **histogram and a Q-Q plot.**

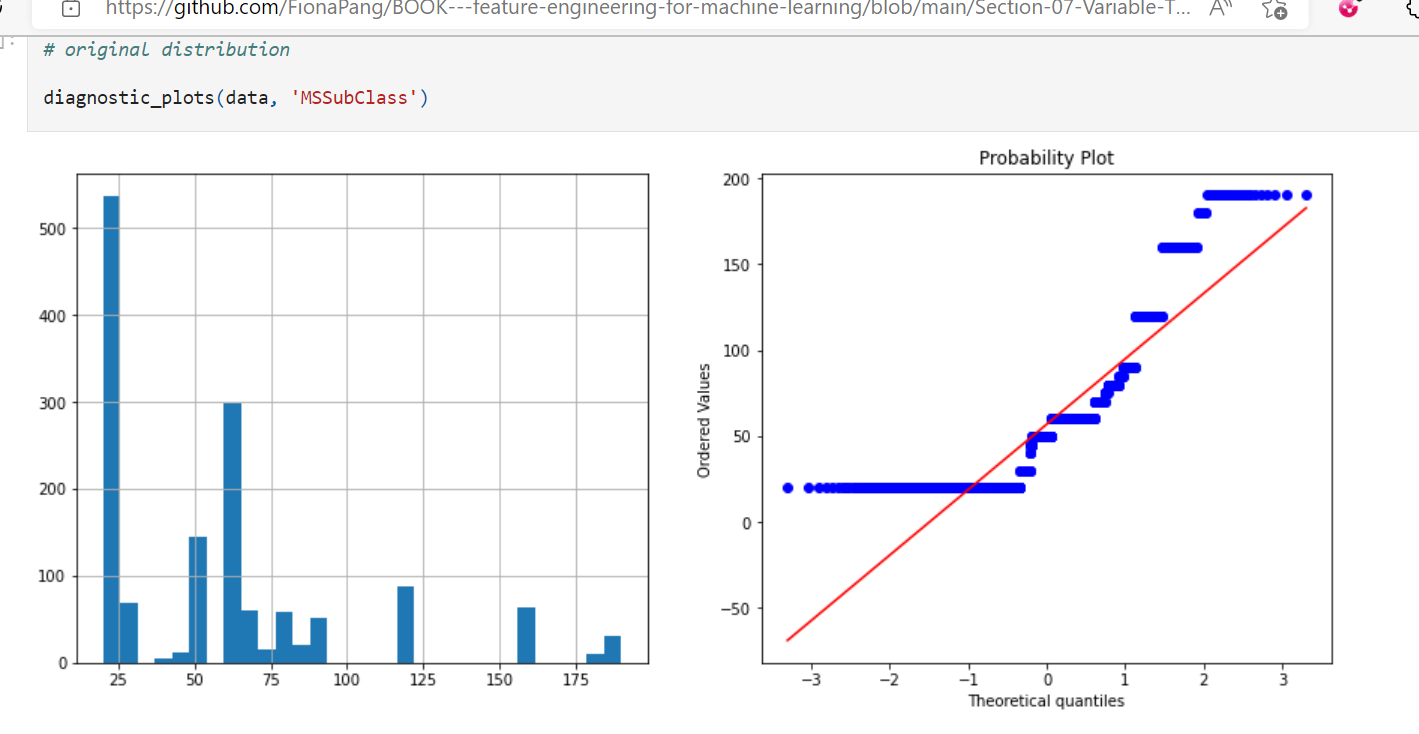
In the **Q-Q pLots**, if the variable is normally distributed, the values of the **variable should fall in a 45** degree line when plotted against the theoretical quantiles. We discussed this extensively in Section 3 of this course.

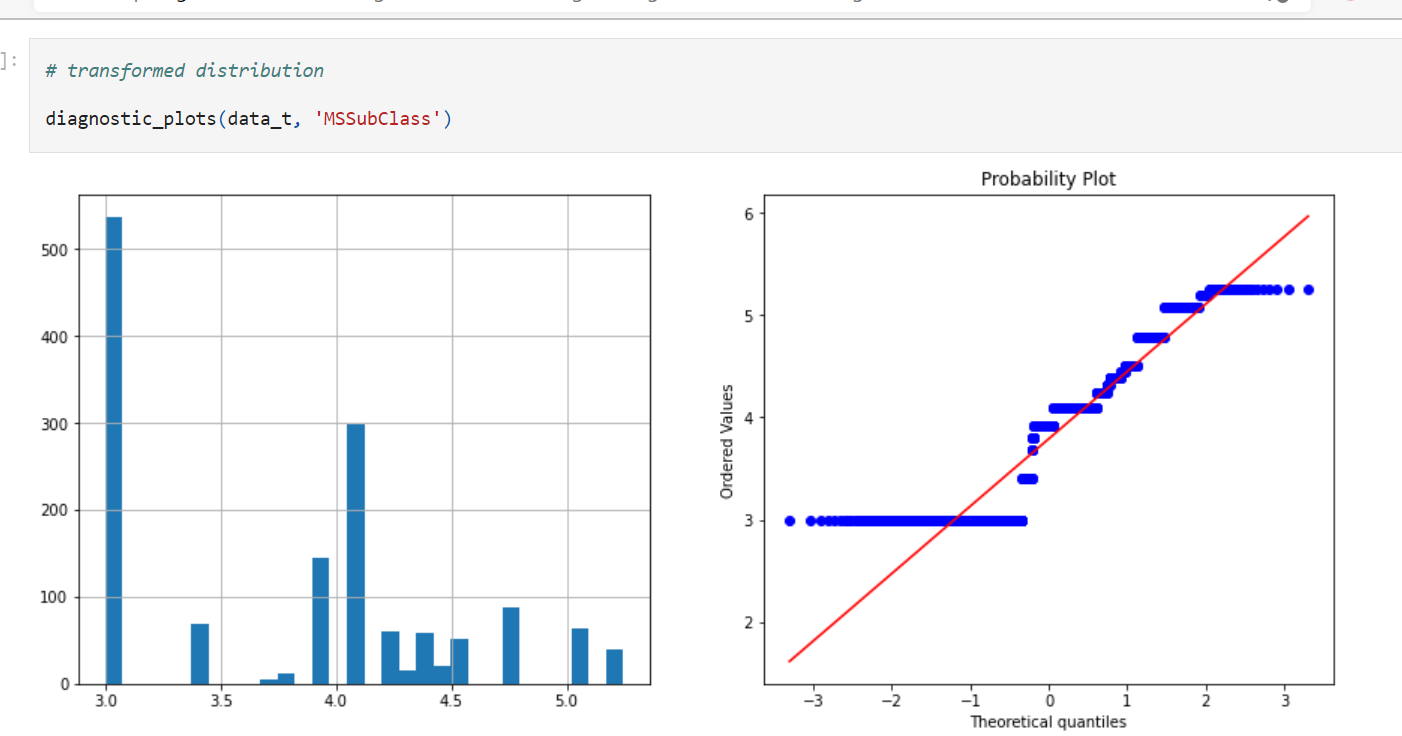


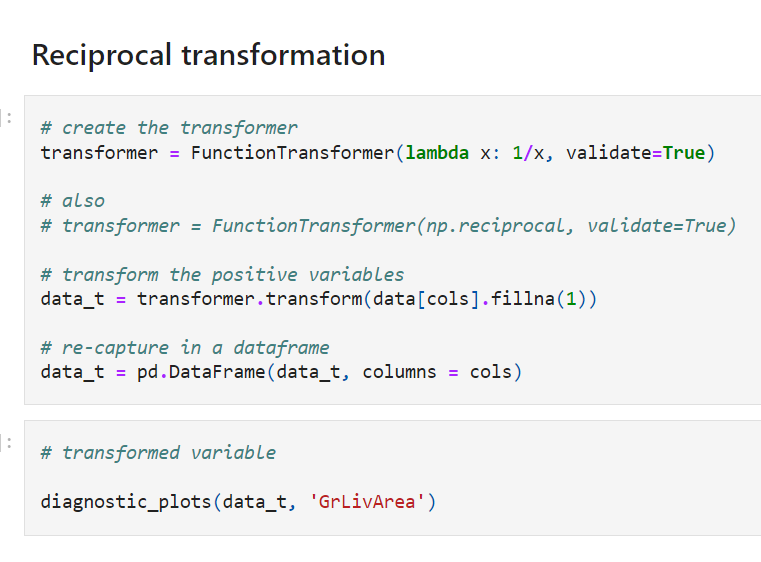


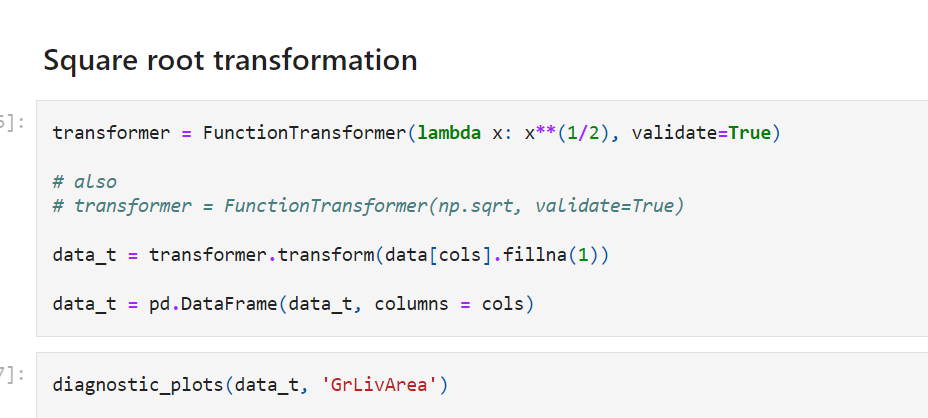


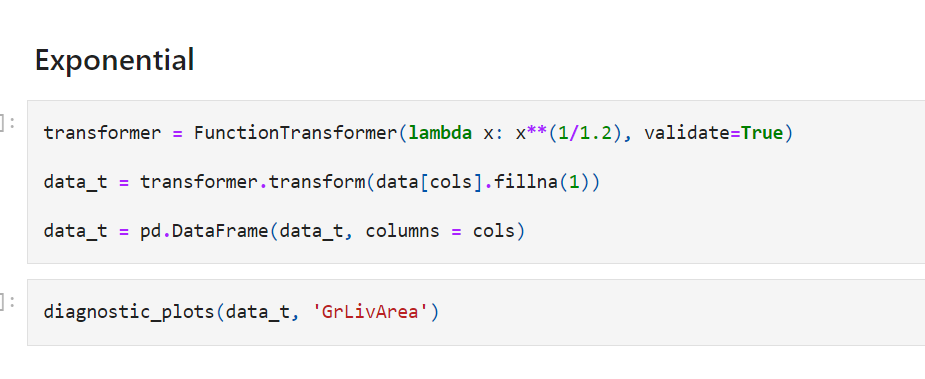


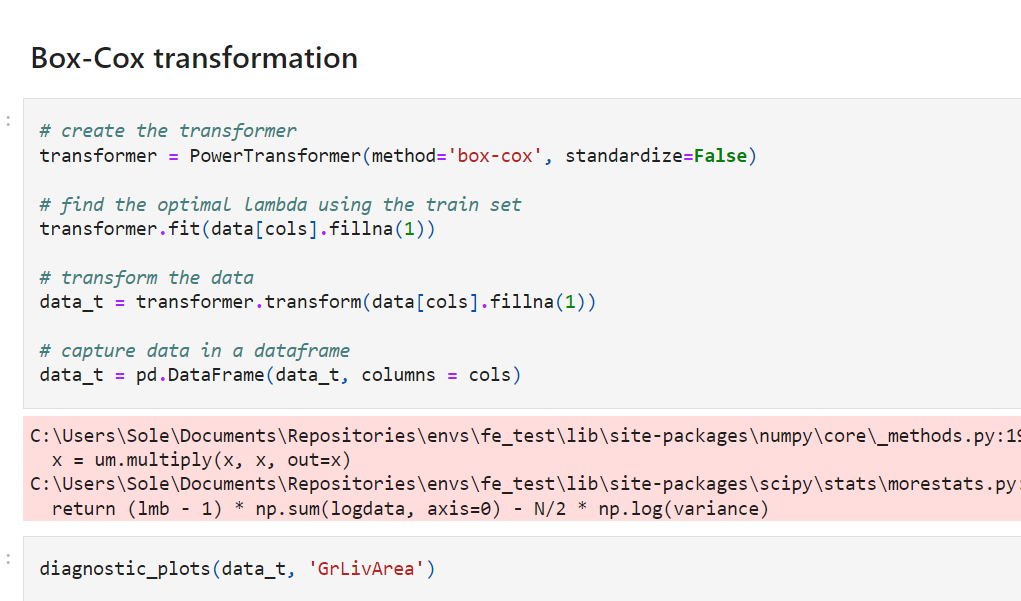














**Gaussian Transformation with Feature-Engine**

Scikit-learn has recently released **transformers to do Gaussian mappings** as they call the variable transformations. The PowerTransformer allows to do Box-Cox and Yeo-Johnson transformation. With the **FunctionTransformer,** we can specify any function we want.

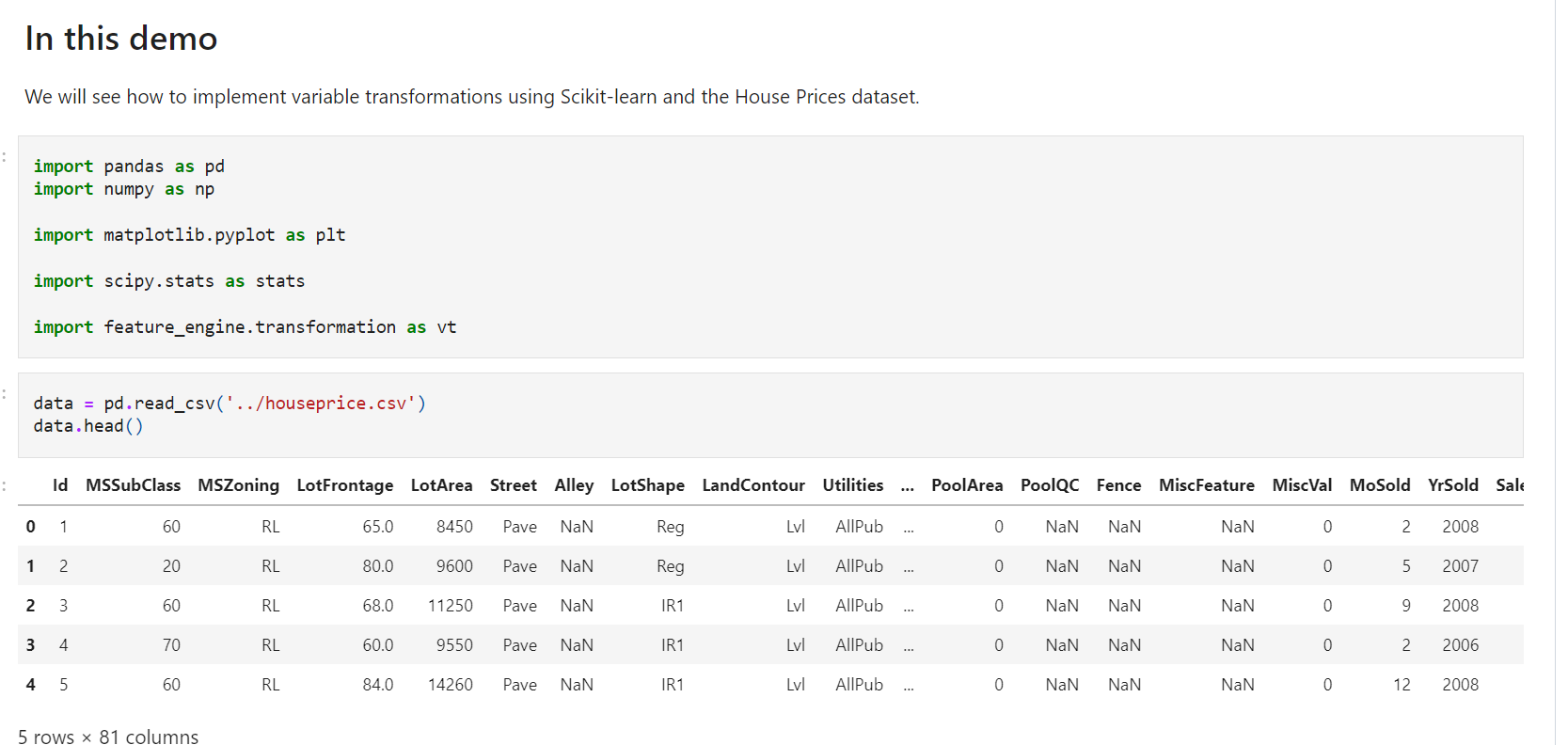
The transformers per se, do not allow to select columns, but we can do so using a **third transformer, the ColumnTransformer**

Another thing to keep in mind is that Scikit-learn transformers **return NumPy arrays**, and not dataframes, so we need to be mindful of the order of the columns not to mess up with our features.

**Important**

**Box-Cox and Yeo-Johnson transformations** need to learn their parameters from the data. Therefore, as always, before attempting any transformation it is important to divide the dataset into train and test set.

In this demo, I will not do so for simplicity, but when using this transformation in your pipelines, please make sure you do so.



**Plots to assess normality**

To visualise the distribution of the variables, we **plot a histogram and a Q-Q plot**. In the **Q-Q pLots,** if the variable is normally distributed, the values of the variable should fall in a 45 degree line when plotted against the theoretical quantiles. We discussed this extensively in Section 3 of this course.

