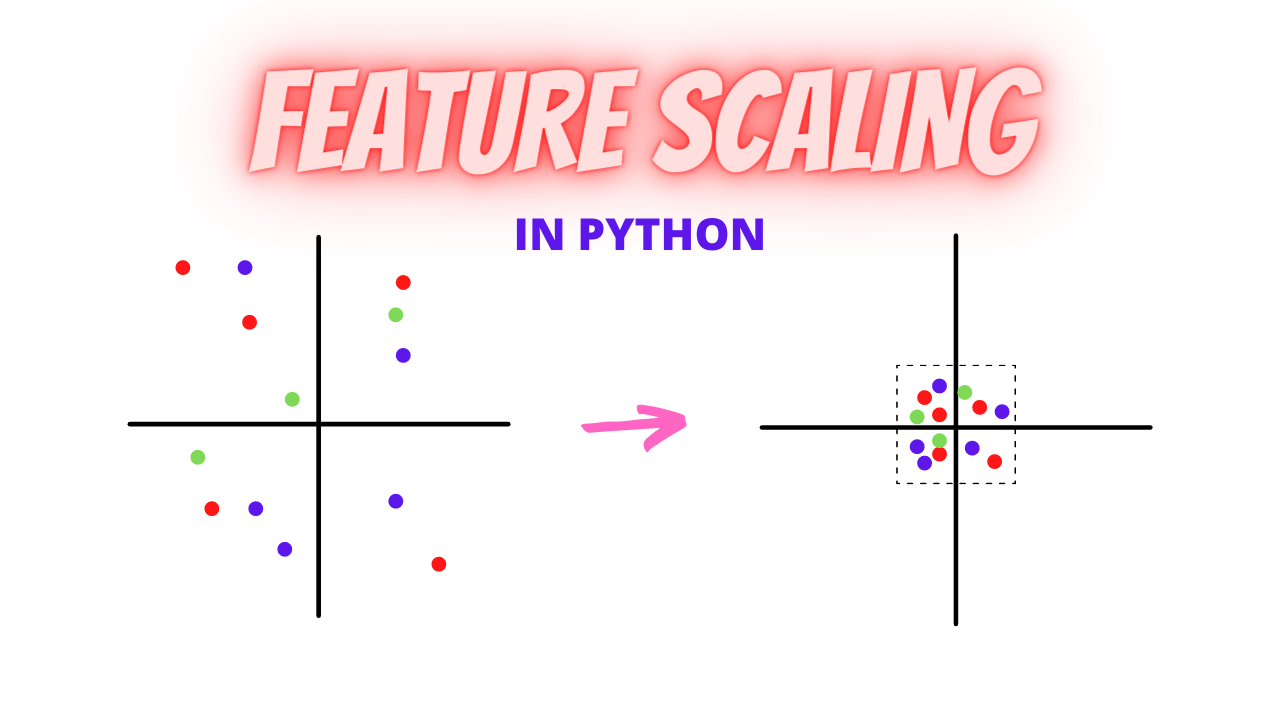
Scaling and Transformation

Feature Scaling is a technique of converting all the values of a feature in the same range. For example, 0 to 1.

Feature Transformation is simply a function that transforms features from one representation to another.

**Why is it required??**

Sometimes we have datasets in which different columns have different units — like one column can be in km, while another can be in meters or centimeters. Or in the same scenario, we have an age column on one side, which ranges from 0 to 1000, and on the other hand, we have a salary column that has all the values greater than 10,000. Then in such a scenario, the difference between these values becomes very large. Due to this difference, **the column having larger values will influence the output more.** Thus we need to perform feature scaling and transformation to make all the values lie in the same range.



**NOTE:**

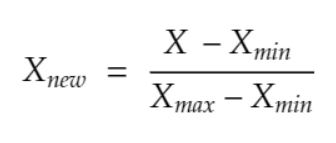
1. Scaler model fitted on the train data will be used to transform the test set. Never fit scaler again on the test data.
2. If dependent features are transformed to normality, Scaling should be applied after transformation.
3. Some algorithms are independent of Scaling. Entropy & Information Gain based techniques are not sensitive to monotonic transformation.Tree-Based Algorithms, Decision Tree, Random Forest, Boosted Trees(GBM, light GBM, xgboost)may not benefit from scaling*.*

**Scaling techniques:**

1. Min Max Scaler
2. Standard Scaler
3. Normalizer
4. Robust Scaler

**1. Min Max Scaler:**

**Min-max scaler** should be the first choice for scaling. For each feature, each value is subtracted by the minimum value of the respective feature and then divide by the range of original maximum and minimum of the same feature. It has a default range between [0,1].



x\_scaled = value of col after scaling  
x = original value  
x\_min = minimum value in the column  
x\_max = maximum value in the column

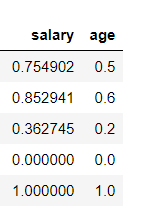
### TO REMOVE UNECESSARY WARNINGS #####  
import warnings  
warnings.filterwarnings('ignore')

import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd

%matplotlib inline

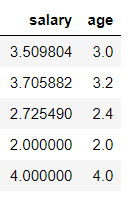
df = pd.DataFrame({  
 'salary':[18000,20000,10000,2600,23000],  
 'age':[23,24,20,18,28],  
 'department':['HR','Marketing','Development','Managment','Legal']  
})  
df.head()

**### Defining Scaler ###**  
from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler()**#### MinMax Scaling ###**col\_names = ['salary', 'age']  
features = df[col\_names]  
features[col\_names] = scaler.fit\_transform(features.values)  
features

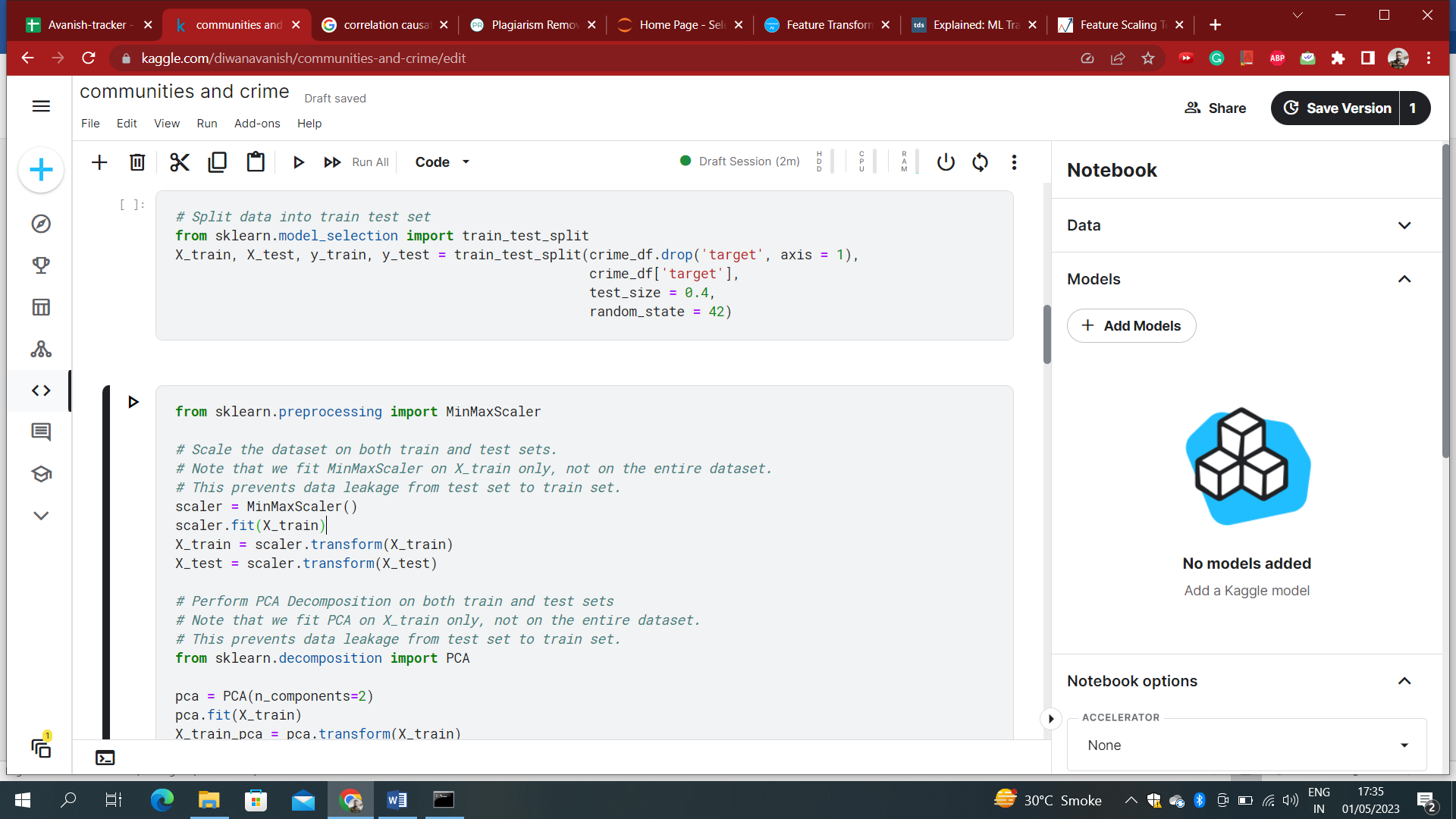


Min Max Scaler will scale down all the values between 0 to 1. If we want the values to be scale down in a custom range, then we can define our own range using **feature\_range**.

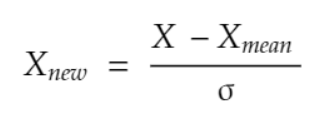
### Defining Scaler ###  
from sklearn.preprocessing import **MinMaxScaler**  
**scaler = MinMaxScaler(feature\_range=(2,4)) ##Custom Scaling Range**#### MinMax Scaling ###  
col\_names = ['salary', 'age']  
features = df[col\_names]  
features[col\_names] = scaler.fit\_transform(features.values)  
features



Note: Check below how scaling is applied for the dataset divided into train and test set.

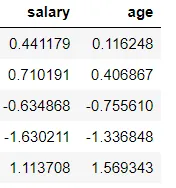


**2. Standard Scaler:**

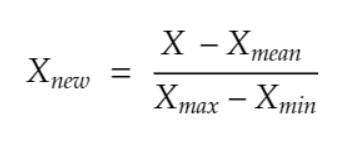
In standardization, we calculate the z-value for each of the data points and replaces those with these values. This will make sure that all the features are centered around the mean value with a standard deviation value of 1. This is the best to use if your feature is normally distributed like salary or age. 

Standard Scaler is also one of the most used and simpler to understand scaler. Rather than scaling values between 0 to 1, it scales the values in a way that mean is 0 and variance is 1.

from sklearn.preprocessing import **StandardScaler**  
scaler = StandardScaler()col\_names = ['salary', 'age']  
features = df[col\_names]  
features[col\_names] = scaler.fit\_transform(features.values)  
features



**3. Normalizer:**

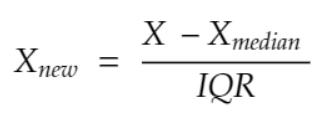
Instead of using the min() value in the previous case, in this case, we will be using the average() value. In scaling, the range of data is changed while in normalization the shape of the distribution of the data is changed.

**4.** **Robust scaling:**

In this method, you need to subtract all the data points with the median value and then divide it by the Inter

Quartile Range(IQR) value. IQR is the distance between the 25th percentile point and the 50th percentile

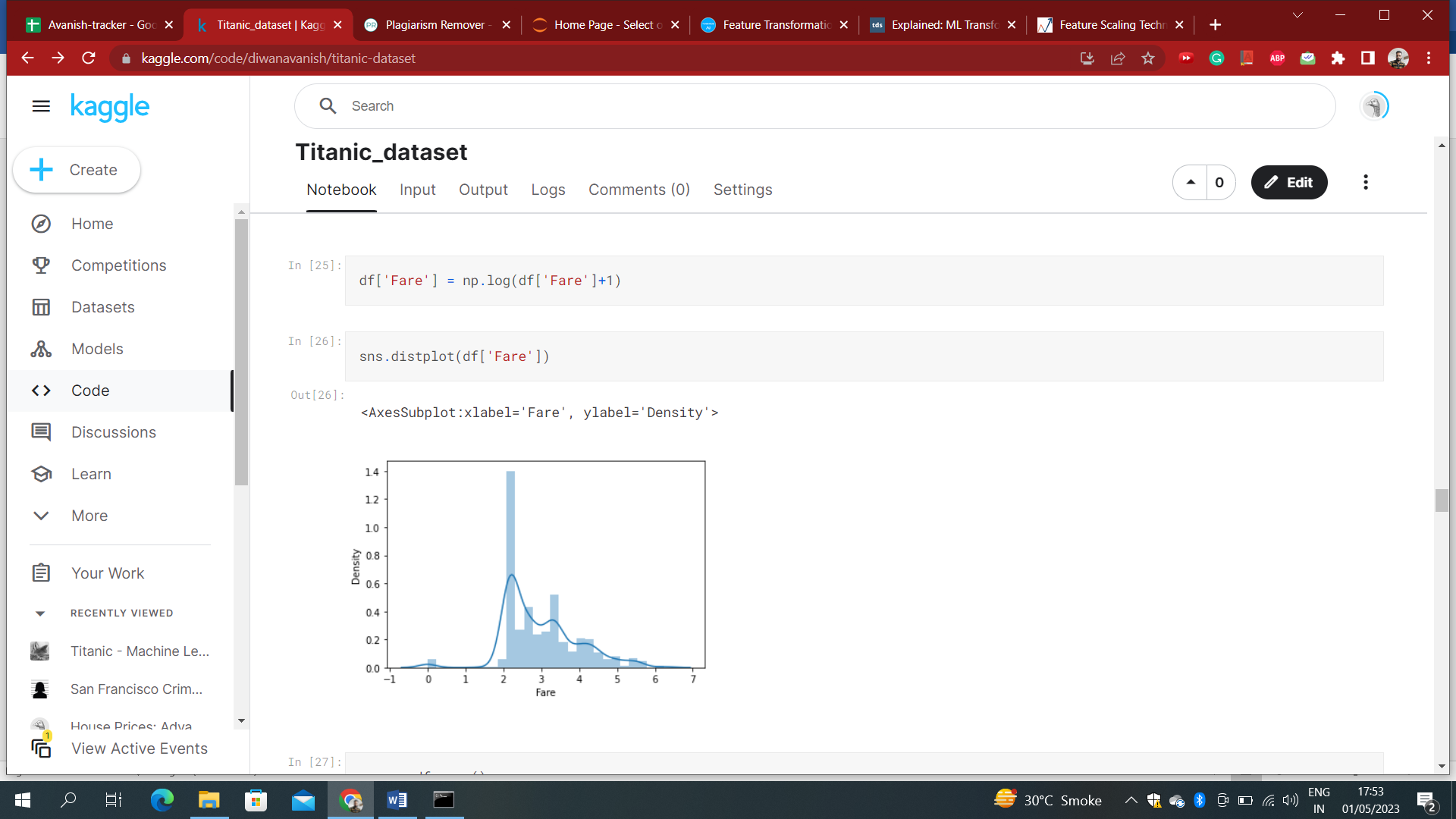
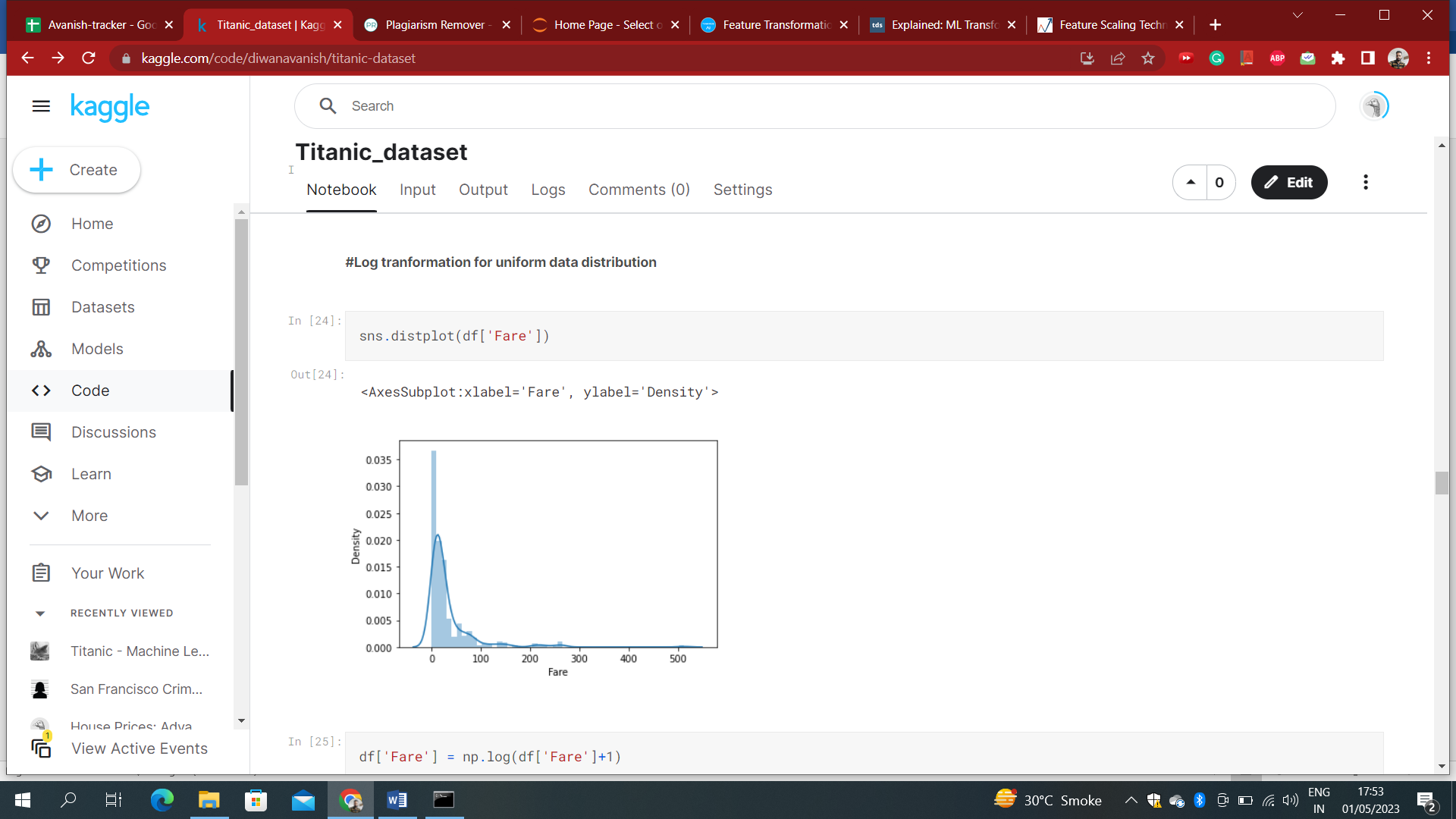
point. This method centres the median value at zero and this method is robust to outliers.



**Transformation techniques:**

1. Log Transformation
2. Box-cox transformation
3. **Log transformation:**

**It is used to convert a skewed distribution to a normal distribution or less-skewed distribution.**In this technique, we take the log of values of the column and take them as a column instead.

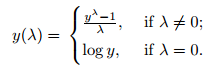
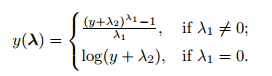


Left Skewed Data Normal Distributed Data

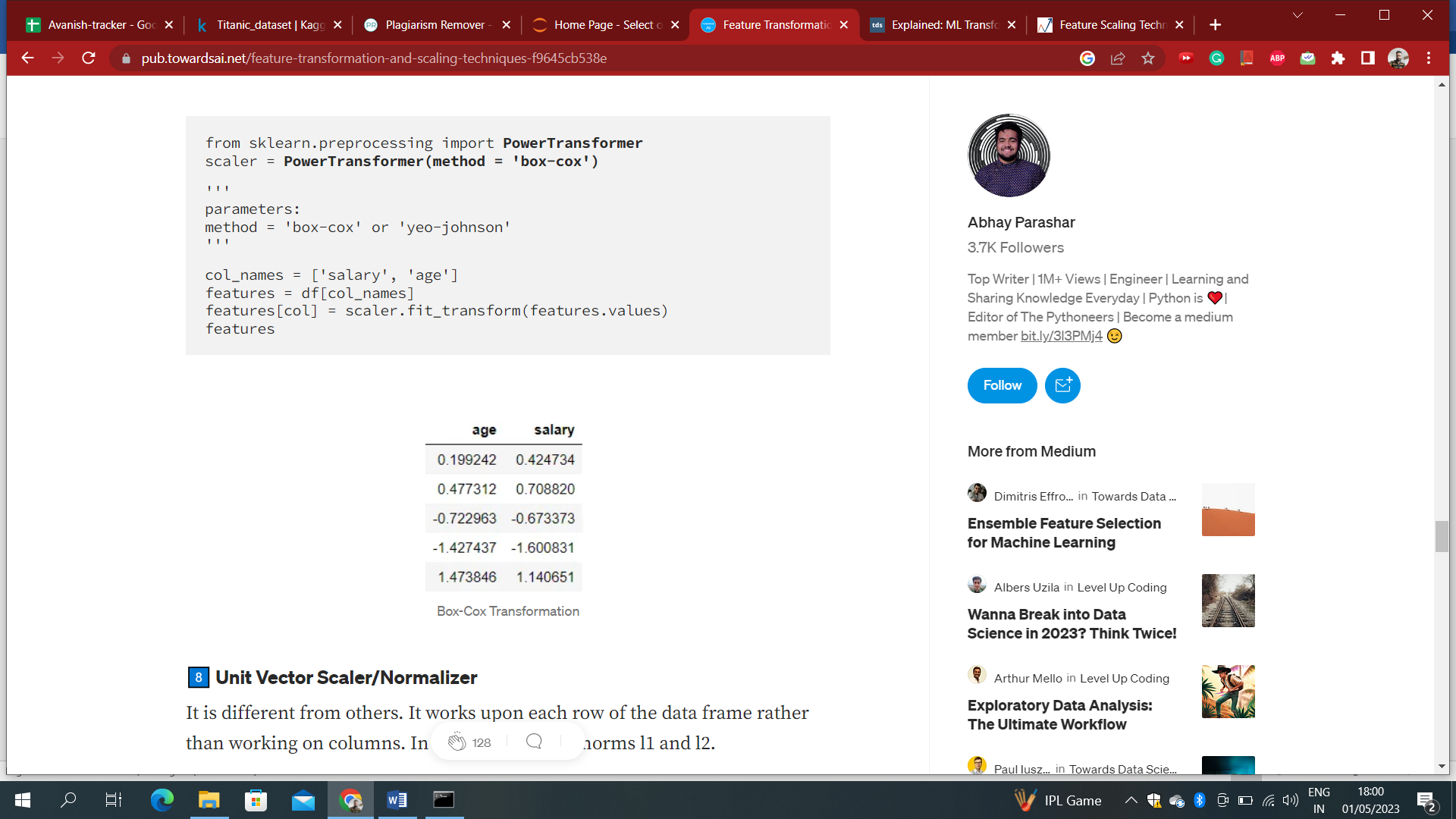
The above Plot Shows How Using Log Transformer Distribution is changed to normal distribution.

1. **Box-cox transformation:**

It also changes the distribution of the variable into a more Gaussian(normal) distribution.

At the core of the Box Cox transformation is an exponent, lambda (λ), which varies from -5 to 5. All values of λ are considered and the optimal value for your data is selected; The “optimal value” is the one which results in the best approximation of a [normal distribution curve](https://www.statisticshowto.com/probability-and-statistics/normal-distributions/). The transformation of Y has the form:  
[](https://www.statisticshowto.com/wp-content/uploads/2015/07/boxcox-formula-1.png)  
  
  
This test only works for positive data. However, Box and Cox did propose a second formula that can be used for negative y-values:  
[](https://www.statisticshowto.com/wp-content/uploads/2015/07/boxcox-formula2.png)

The formulae are deceptively simple. Testing all possible values by hand is unnecessarily labor intensive; most software packages will include an option for a Box Cox transformation.

****