# Some Data Science Terminologies

This file is a description of some data science terminologies with reference to the file

Predicting Wine Quality.ipynb

## **Data distribution**:

Data distribution is a representation of how the data is distributed within the limits of its values and the frequency of each value. There are few standard statistical distributions:

*Uniform distribution*: Dataset is equally distributed.
eg) [1,2,3,4,5,6,7,8,9,10]
Here, all the values of data set occur only once and so we term it as uniform distribution.

*Normal distribution*: When the data set is concentrated near the mean and is meagre when moving away from the mean. This is the common type of distribution.

Representation:

*Basic Statistics:*
A data distribution can be represented with its mean, standard deviation and range.
In python, statistical distribution of a data frame can be obtained from a method called 'describe()'

In the file (3rd Cell):

Wdata.describe()

Gives the count (number of values), mean, std (standard deviation), min, max and percentile values for 25, 50, 75.
Note: X percentile of a dataset is the value that is greater than the X% of the dataset. Zero percentile is min, 100(approx) percentile is maximum and 50 percentile is median.

*Histogram*:
The distribution of a dataset can be visualized in the form of histogram. X axis of hisogram contains the range of values of the dataset, divided equally into a number of bins (intervals). Y axis has the frequency of each bin.
The distribution curve with histogram was obtained in the file (4th cell) using the distplot function from seaborn library:

sns.diatplot(a)

where *a* is the column to be explored.

## **Correlation**:

Correlation is the measure of linear relationship between two variables. Correlation is defined to be in a range of (-1, 1). Here, sign is the measure of direction of relation and magnitude is the measure of intensity of relation. Correlation of 1 implies there is a direct linear relationship between the two variables and correlation of -1 implies there is an inverse linear relationship between the two. A value of 0 means there is no relationship between the two variables.

Correlation between two variables is obtained by:

$$\frac{1}{n-1}.\frac{\sum_x \sum_y (x-\bar{x})(y-\bar{y})}{S_x.S_y}$$

Where n – the total number of values

$\bar{x}$ – Mean of x (first variable)

$\bar{y}$ – Mean of y (second variable)

$S_x$ – Standard deviation of x

$S_y$ – Standard deviation of y

Basically we find correlation matrix to reduce dimensions. We can remove a column from the dataset if it has very less correlations with the target variable and also with the other variables. In that case the column does not have a significant role in affecting the prediction.

In the file (8th cell),

We have found the correlation matrix of the dataframe Wdata using corr() method.

```
Wdata.corr()
```

We make use of heat map from seaborn to plot this correlation.

```
sns.heatmap()
```

Here we note that 'fixed acidity', 'residual acidity', 'free sulfur dioxide', 'pH' do not have a significant correlation with 'quality' ( which is our target variable).

But also note that the correlation of the above mentioned column with the other columns were significant.

'Fixed acidity' has a strong correlation with 'citric acid' and 'density' which have significant correlation with 'quality'.

Similarly 'residual sugar' has strong correlation with density, 'free sulfur dioxide' with 'total sulfur dioxide' and 'pH' has good correlation with citric acid.

Since there is no column that has negligible correlation with all other columns, we cannot remove any column. So we could not reduce dimension in this way.

## Reducing the dimensionality:

In the above section there was an attempt to reduce the dimensionality but what is the need to reduce the dimensionality? In what way higher dimension will affect the problem?

We have 11 variables (columns) without including the target variable. What we are going to deal with is 11 Dimensional problems. At such high dimensions our space will become sparse. Though there are 11 variables, the number of instances (rows) is only around 1600. So, most of the space in the 11-D solution space would be sparse. To deal with 1600 instances we may have to deal with much more than $10^{11}$ instances! If we could reduce the dimension we can actually reduce the computation required.

To be clearer, let us compare a 3D space and a 2D space. Let the space be limited to 10 units. A 3D space has coordinates of (0,0,0),(0,0,1),....,(0,1,0),(0,1,1)...,(1,0,0),(1,0,1),....(10,10,10). It has 1000 possible points. A 2D space with 10 units and coordinates (0,0),(0,1),...(1,0),(1,1),...(10,10). It has 100 possible points. Now say we have 10 instances in our data set. If we represent them in the 2D space we have filled 10% of the solution space. If we represent them in a 3D space then we would have filled only 1% of the whole space. 99% of the 3D space is unused! So computation becomes heavy!

So reducing the dimension will increase the computability but it would affect the solution. Actually reducing dimension is trade-off between computability and accuracy. Another problem in this thread is which variable to remove? This is depends on the correlation of the variables with the other variables and that is the reason we were dealing with correlation matrix.

## Principal Component Analysis:

Principal Component analysis is another way of reducing the dimension. What is meant by 'Principal Components'? These are another set of variables we will change to, from our present set of variables. We have found that in this problem we could not reduce the dimension with the present of variables (we could not drop any of them). So we can change the variable set to their principal components.

Now are we changing the variables? Doesn't this change the problem? No it doesn't because the principal components are linear combinations of the present set of variables. Now let our variable be a, b, c, d, e and f then we will have 6 principal components p, q, r, s, t and u such that

$$p = Aa + Bb + Cc + Dc + Ee + Ff$$
$$q = Aa + Bb + Cc + Dc + Ee + Ff$$
$$r = Aa + Bb + Cc + Dc + Ee + Ff$$
.
.

Where A, B, C, D, E and F are arbitrary coefficients and *are different for different principal components*. We choose A, B, C, D, E and F in such a way that the correlation between the principal coefficients is 0. But how do we choose them in a way the correlation between the principal components are zero?

We can perform eigen value decomposition and the eigen vector matrix will give A, B, C, D, E and F for all six rows and eigen vector will give how the significant the eigen values are. If we find the value of the eigen value to be negligible then we can remove that principal component there by reducing dimension.

It has been a lot of time since we were speaking about the problem. So returning to Principal component Analysis section in the jupyter notebook, we have done the principal component analysis for X (set of 11 variables).

Firstly we have standardized the matrix so that the mean of each column is equated to zero and the distribution becomes more like a normal distribution.

(Cell 16th)Then we make PCA module from scikit learn decomposition. Before we do PCA we obtain the cumulative sum of the variance ratio. When we plot it against the number of component we can find how many components can be used to represent significant information. Yes cumulative sum of variance is the measure of the amount of information.

Now we find that 7 principal components are enough to obtained sufficient information from the matrix. (See output off cell 16) these 7 components are those which have highest value of eigen values. Then we can do the principal component analysis by

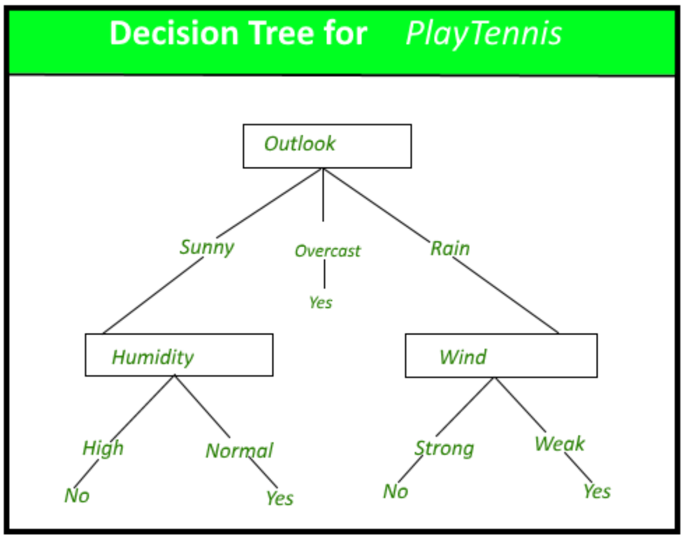
PCA()

We now save them as X_sk which is the dimension reduced matrix of size 1593 X 7. Later in the last part of the notebook when we fit the models with the principal component we found that the error seemed to be increased. This is because the reduction in dimension will affect the accuracy as discussed earlier.

## **Decision Tree Regressor**

Decision tree is actually the way we would make decision. If want we want to play tennis during a rainy season, we look out for weather. In case it is Sunny we would check humidity else if it is rainy we will see whether it is windy. If it either of the case we would make another decision. So our decisions will vary based on certain variable and then further decision makings are made based on other variables. We can make these decision flows in the form of a tree and such a tree would be called decision tree.

Below is the decision tree to decide whether we would like to go to play tennis or not. [img source : geeksforgeeks.org]



We can perform regression on a Decision tree. We can construct a decision tree from the dataset and perform regression on it. How do we construct a decision tree? We have to first decide which variable we have to make decision first. The first variable to make decision on will depend on the maximum information gain, minimum entropy loss. This algorithm is known as id3 algorithm.

We have performed Decision tree regression on the given dataset using DecisionTreeRegressor function from scikit learn’s tree module.

Before going to the practical implementation of Decision Tree Regressor, let us review about model selection. Model selection is to split the given dataset into training and testing dataset. So that the model can be fitted with the training dataset and the predicted value from the test data can be analysed with the actual test target data and the accuracy or the error of the model can be constructor. Throughout the notebook we have calculated the mean absolute error and root mean square error. We would review these concepts very soon.

DecisionTreeRegressor builds a decision tree model. The model can be fitted with train dataset. Test data is implemented to predict via predict () function and it is tested for error.

## Linear Regressor:

Linear Regressor model is making a linear relationship between the target variable and the other variables. If the target variable is T and other variables are a, b, c, d, e and f, the linear regression would construct a model of the form:

T= Aa + Bb + Cc + Dd + Ee + Ff, where

A, B, C, D, E and F are constants. How do we find the value of these constants?

We can make use of gradient descent algorithm or other advanced algorithms to find the same. In the notebook, we make use of Linear regression from scikit learn's Linear model.

## Random Forest:

We have seen tree and it's time for forest. Forest is of course collection of trees. There are many decision trees in this random forest and these trees are constructed by dropping some random variables. The results of individual trees are combined to give the result for the random forest.

In the notebook, Random forest has been implemented by sckit learn. Random forest Regressor from the module ensemble is used for this purpose.

## Mean Absolute Error:

We have calculated the mean absolute error for all the models. Now let us see what Mean Absolute error is. Mean absolute error is intuitive. As the name suggests it is the mean of the absolute value of the errors.

MAE = mean ($|y_{pred} - y_{act}|$)

It is the measure of magnitude of deviation of predicted value from the actual value and it does not take into account the sign. For a most suitable model it would be 0.

## Root Mean Square Error:

We have also calculated Root mean Square error for all models. Root mean square error is the root of sum of squares of the errors.

$$\text{RMSE} = \sqrt{\frac{\sum_{n}(y_{\mathrm{Pr}ed} - y_{act})^2}{n}}$$

It is similar to mean absolute error. It is a quadratic version of the mean square error. If we have a larger error, it gets squared and increases the RMSE more vigorously than MAE. So difference between MAE and RMSE can be considered as the measure of variances. RMSE will always be greater than or equal to the MAE. And both can range from 0 to infinity.