**General Info Page.**

**Boxcox transformation:**

**Why is it used:** It is basically used to remove skewness in the data and make the data normal. Important to have in certain kinds of statistical tests

**Peculiarities:**

Difference between boxcox and boxcox1p is as follows

Boxcox func: (x \*\* lmbda - 1) / lmbda if lmbda != 0

Log(x) if lmbda == 0

Boxcox1p func: ((1 + x) \*\* lmda - 1) / lmbda if lmbda != 0

Log(1 + x) if lmbda == 0

**References:**

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.boxcox.html>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.boxcox1p.html>

**Lasso Regression:** is a regression analysis method that performs both variable selection and regularization. For Python’s sklearn It’s a linear model trained with L1 prior as regularizer.

**Function:** Minimizes (1 / (2 \* n\_samples)) \* ||y - Xw||^2\_2 + alpha \* ||w||\_1

**References:**

<http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html>

<https://en.wikipedia.org/wiki/Lasso_(statistics)>

**Elastic Net Regression:** similar to Lasso regression with a difference that it is Linear regression with both L1 and L2 priors for regularization.

**Function:**

* 1 / (2 \* n\_samples) \* ||y - Xw||^2\_2 + alpha \* l1\_ratio \* ||w||\_1 + 0.5 \* alpha \* (1 -l1\_ratio) \* ||w||^2\_2

You can control the L1 and L2 regularization as follows

* a \* L1 + b \* L2

where

* alpha = a + b and l1\_ratio = a / (a + b)

**References:**

<http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html>

**Kernel trick:**

**Why is it used:** Any linear model can be converted into a non linear model using the kernel trick replacing its features by the kernel function. Kernel function allow the model to operate in higher dimension spaces without needing to compute the actual co-ordinate of the data in the higher dimensional space. They achieve to do so by taking inner products of the images of data in the feature space.

**References:**

<https://en.wikipedia.org/wiki/Kernel_method>

**Kernel Regression:** Combines linear regression with the kernel trick. i.e. learns a linear function in the higher dimensional space defined by the kernel function or learns a non-linear function in the original feature space.

**References:**

<http://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html>

**Gradient Boosting:** Boosting is basically a method to make an ensemble of a number of weak learners typically decision trees. The models can be imagined as stacked over each other taking as input the residual of the previous model and trying to reduce it in the combined model.

**Functional form:**

_{m+1}(x) = F_m(x) + h_m(x) = y, for  \geq 0

**References:**

<http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/>

<https://en.wikipedia.org/wiki/Gradient_boosting>

**XGBoost:** is an advanced implementation of the gradient boosting algorithm optimized for performance and computation.

**References:**

<https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>

**How to install**: XGBoost in anaconda

<https://anaconda.org/conda-forge/xgboost>

**Light Gradient Boosting:** is a fast, distributed, high performance gradient boosting algorithm framework. It is based on decision trees. It splits the tree leaf-wise rather than depth-wise or level wise.

**References:**

<https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/>

**How to install:** LightGBM in anaconda

<https://anaconda.org/conda-forge/lightgbm>