**Machine Learning : 'Regression' - Day 3**

* Published on Published onFebruary 28, 2018

[[](https://www.linkedin.com/in/panchalshivam/)](https://www.linkedin.com/in/panchalshivam/)

**[Shivam Panchal](https://www.linkedin.com/in/panchalshivam/)**

 8

*  11

[Write an article](https://www.linkedin.com/post/new)

In the last to last post, we discussed about what is Regression and in the last one, we talked about the assumptions or so called limitations of linear regression.

Linear Regression is assumed to be the simplest machine learning algorithm the world has ever seen, and yes! it is!

Anyways! we also discussed how your model can give you poor predictions in real time if you don't obey the assumptions of linear regression. Whatever you are going to predict, whether it is stock value, sales or some revenue, linear regression must be handled with care if you want to get best values from it.

Linear regression says, the data should be linear in nature, there must be a linear relationship. But, wait! the real world data is always non-linear. Yes, so, what we should we do, should we try to bring non-linearity into the regression model, or check out the residuals and fitted values, keep applying transformations and working harder and harder to get the best predictive model using linear regression.

Now, the question is, should it be considered as the solution or is there any other way to deal with this, so that I can get a better predictive model without getting into these assumptions of linear regression.

Yes! there is a solution, in fact a bunch of solutions.

There are many different analytic procedures for fitting regressive models of nonlinear nature (e.g., Generalized Linear/Nonlinear Models (GLZ), Generalized Additive Models (GAM), etc.), or more better models called tree based regressive models!

Most of us know about Random Forest and Decision Trees, it is very common, in case of classification or regression, they often perform far better than other models.

In this post, I will be mainly talking out tree based models such as Decision Trees and ensemble tree based like Random forests. Tree based model have proven themselves to be both reliable and effective, and are now part of any modern predictive modeler’s toolkit.

But, there are some cases when a linear regression is assumed better than a tree based models like the following cases.

1. *When the underlying function is truly linear*
2. *When there are a very large number of features, especially with very low signal to noise ratio. Tree based model have a little trouble modeling linear combinations of a large number of features.*

The point is: there are probably only a few cases in which linear models like SLR is better than tree based models or other non-linear models as these fits the data better from the get-go without transforms.

They’re more forgiving in almost every way. You don’t need to scale your data, you don’t need to do any monotonic transformations (log square root etc). You often don’t even need to remove outliers. You can throw in features, and it’ll automatically partition the data if it aids the fit. You don’t have to spend any time generating interaction terms as in case of linear models. And perhaps most important: in most cases, it’ll probably be notably more accurate.

**The bottom line is:** *You can spend 3 hours playing with the data, generating features and interaction variables and get a 77% r-squared; and I can “from sklearn.ensemble import RandomForestRegressor” and in 3 minutes get an 82% r-squared.*

I am not creating a hype for these tree model, I am going to show you that below in an implementation with python and same housing data which we have been using since last two posts.

Let me explain it using some examples for clear intuition with an example.

Linear regression is a linear model, which means it works really nicely when the data has a linear shape. But, when the data has a non-linear shape, then a linear model cannot capture the non-linear features. So in this case, you can use the decision trees, which do a better job at capturing the non-linearity in the data by dividing the space into smaller sub-spaces depending on the questions asked.

Now, the question is when do you use linear regression vs Decision Trees? I guess the Quora answer here would do a better job than me, at explaining the difference between them and their applications. Let me quote that for you:

Let’s suppose you are trying to predict income.

The predictor variables that are available are education, age, and city.

Now in a linear regression model, you have an equation with these three attributes. Fine. You’d expect higher degrees of education, higher “age” and larger cities to be associated with higher income.

But what about a PhD who is 40 years old and living in Scranton Pennsylvania? Is he likely to earn more than a BS holder who is 35 and living in Upper West SIde NYC? Maybe not. Maybe education totally loses its predictive power in a city like Scranton? Maybe age is a very ineffective, weak variable in a city like NYC?

This is where decision trees are handy. The tree can split by city and you get to use a different set of variables for each city. Maybe Age will be a strong second-level split variable in Scranton, but it might not feature at all in the NYC branch of the tree. Education may be a stronger variable in NYC.

Decision Trees, be it Random Forest or GBM, handle messier data and messier relationships better than regression models. ANd there is seldom a dataset in the real world where relationships are not messy. No wonder you will seldom see a linear regression model outperforming RF or GBM

So, this is the main idea behind tree (Decision Tree Regression) and ensemble based models (Random Forest Regression/Gradient Boosting Regression/ Extreme Boosting Regression).

The following is the link to repo. Check out the Day 3 implementation.

I used the Boston Housing Data to train all the different available regressive models present in the sklearn.

These were my results for R-squared on train and testing data (30%). You can find the following results there. The repo is open to contribute, I invite fresh buddies to improve my model there, apply some feature engineering, or do some hyper-parameter tuning using Grid Search to make it more better.

As you can GBM/RF are performing the best.

Below are the links to almost all the regression techniques in sci-kit learn.

1. **Ordinary least squares Linear Regression:**

**2. Linear least squares with l2 regularization:**

http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.Ridge.html

**3. Linear Model trained with L1 prior as regularizer (aka the Lasso):**

http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.Lasso.html

**4. Support Vector Regression:**

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

**5. A decision tree classifier:**

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

**6. A random forest regressor.**

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

**7. Linear model fitted by minimizing a regularized empirical loss with SGD:**

SGD stands for Stochastic Gradient Descent: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate).

http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.SGDRegressor.html

**8. Gradient Boosting for regression:**

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

Feel Free to Criticize!

See you in the next!!