**Building and evaluating Models**

**Prediction**

Predictive modeling cmes under the section of supervised learning of Machine Learning. To implement predictive modeling in python we use scikit-learn.

Steps followed for performing Prediction modeling:

1. Descriptive analysis on the Data – 50% time
2. Data treatment (Missing value and outlier fixing) – 40% time
3. Data Modelling – 4% time
4. Estimation of performance – 6% time

**Stage 1: Descriptive Analysis / Data Exploration:**

* Identify ID, Input and Target features
* Identify categorical and numerical features
* Identify columns with missing values

We have used various features selection techniques provided by scikit-learn.

1. Recursive FeatureElimination
2. select K Best Features
3. Lasso Lars Regression Model
4. Reidge Regression Model
5. Linear Regression Model

**Stage 2: Data Treatment (Missing values treatment):**

There are various ways to deal with it:

* Create dummy flags for missing value(s) : It works, sometimes missing values itself carry a good amount of information.
* Impute missing value with mean/ median/ any other easiest method : Mean and Median imputation performs well, mostly people prefer to impute with mean value but in case of skewed distribution I would suggest you to go with median. Other Intelligent methods are imputing values by similar case mean and median imputation using other relevant features or building a model. For Example: In Titanic survival challenge, you can impute missing values of Age using salutation of passengers name Like “Mr.”, “Miss.”,”Mrs.”,”Master” and others and this has shown good impact on model performance.
* Impute missing value of categorical variable: Create a new level to impute categorical variable so that all missing value is coded as a single value say “New\_Cat” or you can look at the frequency mix and impute the missing value with value having higher frequency.

**Stage 3. Data Modelling:**

We use linear regression, [Random Forest](https://www.analyticsvidhya.com/blog/2015/09/random-forest-algorithm-multiple-challenges/)  and Neural Network algorithms techniques, depending on the business problem.

Data modeling techniques appled:

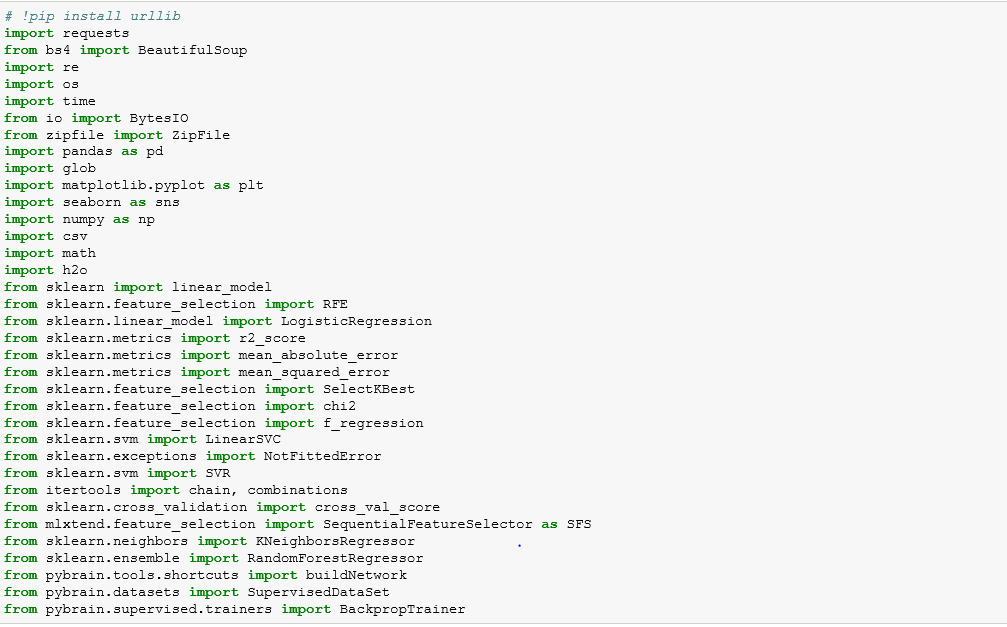
1. Linear Regression
2. Random Forest Regression Algorithm
3. Neural Network Algorithm (Backperformance)
4. H20 Algorithm (Deep Learning)

**Stage 4. Estimation of Performance:**

There are various methods to validate your model performance, you could divide your train data set into Train and validate (ideally 70:30) and build model based on 70% of train data set. Now, cross-validate it using 30% of validate data set and evaluate the performance using evaluation metric.

We apply all the stages of predictive modelling on the freddiemac’s Single family loan origination dataset.

**Step 1: Get all the imports required**



**Step 2: Download data form the freddiemac’s website based on user input.**

* Taking input form the user



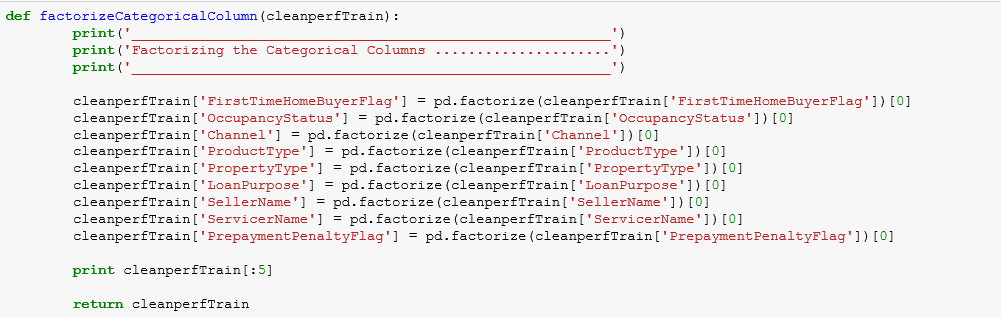
* Downloading data automatically by-passing the login



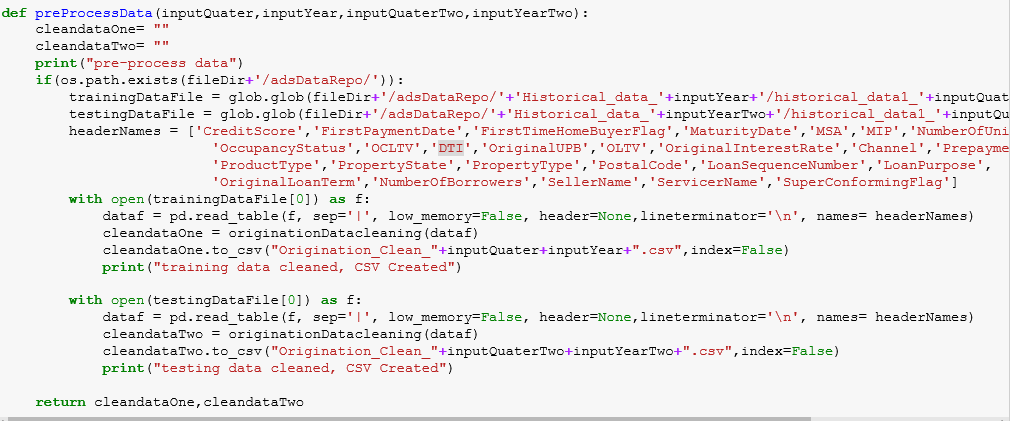
**Step 3: Pre-process the Data**

* Clean data, factorize data and assigning datatypes to the variables.





* Generating clean data fie and saving it on the local



**Step 4: Feature Selection**

The classes in the [sklearn.feature\_selection](http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selection) module can be used for feature selection/dimensionality reduction on sample sets, either to improve estimators’ accuracy scores or to boost their performance on very high-dimensional datasets.

**Removing features with low variance**

[VarianceThreshold](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html#sklearn.feature_selection.VarianceThreshold) is a simple baseline approach to feature selection. It removes all features whose variance doesn’t meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

As an example, suppose that we have a dataset with boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by

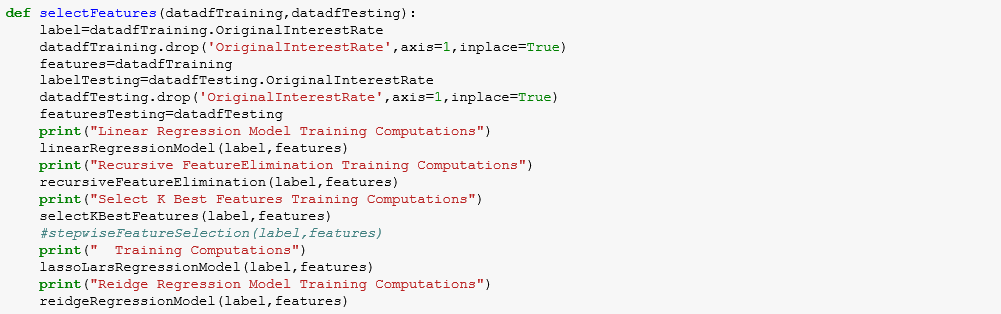
\mathrm{Var}[X] = p(1 - p)

so we can select using the threshold .8 \* (1 - .8).

**Univariate feature selection**

Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a pre-processing step to an estimator. Scikit-learn exposes feature selection routines as objects that implement the transform method:

* [SelectKBest](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest) removes all but the k highest scoring features
* [SelectPercentile](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html#sklearn.feature_selection.SelectPercentile) removes all but a user-specified highest scoring percentage of features
* using common univariate statistical tests for each feature: false positive rate [SelectFpr](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFpr.html#sklearn.feature_selection.SelectFpr), false discovery rate [SelectFdr](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFdr.html#sklearn.feature_selection.SelectFdr), or family wise error [SelectFwe](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFwe.html#sklearn.feature_selection.SelectFwe).
* [GenericUnivariateSelect](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.GenericUnivariateSelect.html#sklearn.feature_selection.GenericUnivariateSelect) allows to perform univariate feature selection with a configurable strategy. This allows to select the best univariate selection strategy with hyper-parameter search estimator.



[**SelectKBest**](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest)

We have implemented [SelectKBest](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest)  algorithm to select our features

Sklearn DOES have a forward selection algorithm, although it isn't called that in scikit-learn. The feature selection method called [F\_regression](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html) in scikit-learn will sequentially include features that improve the model the most, until there are K features in the model (K is an input).

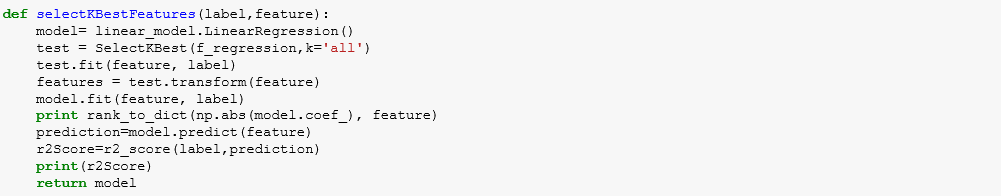
It starts by regression the labels on each feature individually, and then observing which feature improved the model the most using the F-statistic. Then it incorporates the winning feature into the model. Then it iterates through the remaining features to find the next feature which improves the model the most, again using the F-statistic or F test. It does this until there are K features in the model.

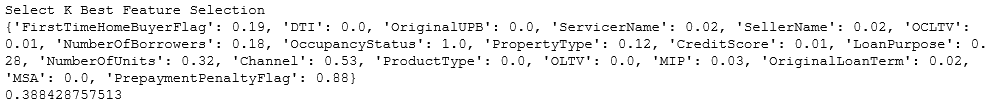
Notice that the remaining features that are correlated to features incorporated into the model will probably not be selected, since they do not correlate with the residuals (although they might correlate well with the labels). This helps guard against multi-collinearity.

These objects take as input a scoring function that returns univariate scores and p-values (or only scores for [SelectKBest](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest)and [SelectPercentile](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectPercentile.html#sklearn.feature_selection.SelectPercentile)):

1. For regression: [f\_regression](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html#sklearn.feature_selection.f_regression), [mutual\_info\_regression](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html#sklearn.feature_selection.mutual_info_regression)
2. For classification: [chi2](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2), [f\_classif](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif), [mutual\_info\_classif](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html#sklearn.feature_selection.mutual_info_classif)

Our code for SelectKBest Features:

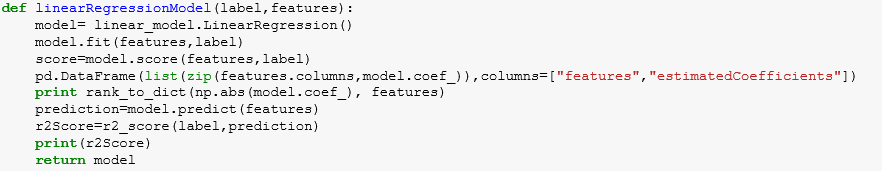


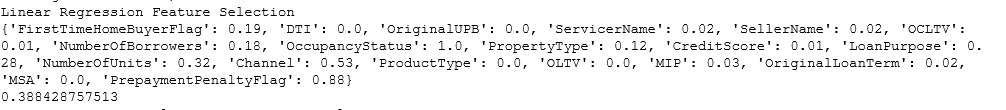


**Linear Regression Model**

[LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression) fits a linear model with coefficients w = (w_1, ..., w_p) to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation. Mathematically it solves a problem of the form:

\underset{w}{min\,} {|| X w - y||_2}^2



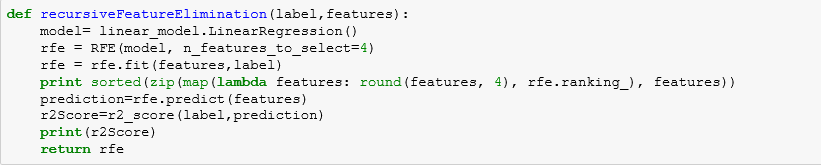


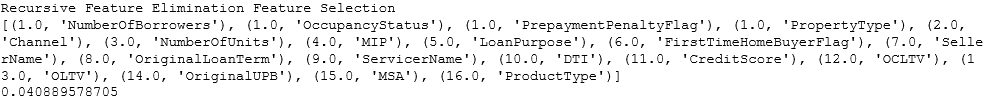
**Recursive Feature Elimination**

Recursive feature elimination is based on the idea to repeatedly construct a model (for example an SVM or a regression model) and choose either the best or worst performing feature (for example based on coefficients), setting the feature aside and then repeating the process with the rest of the features. This process is applied until all features in the dataset are exhausted. Features are then ranked according to when they were eliminated. As such, it is a greedy optimization for finding the best performing subset of features.

The stability of RFE depends heavily on the type of model that is used for feature ranking at each iteration. Just as non-regularized regression can be unstable, so can RFE when utilizing it, while using ridge regression can provide more stable results.

Sklearn provides [RFE](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html) for recursive feature elimination and [RFECV](http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html) for finding the ranks together with optimal number of features via a cross validation loop.





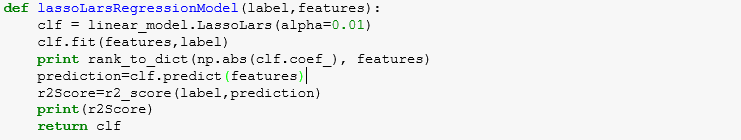
**Lasso lars**

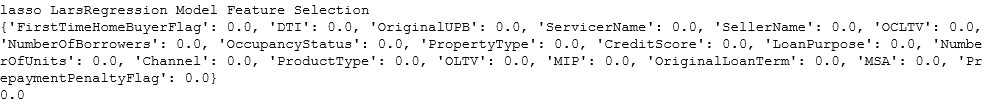
Lasso model fit with Least Angle Regression a.k.a. Lars. It is a Linear Model trained with an L1 prior as regularizer.

Least-angle regression (LARS) is a regression algorithm for high-dimensional data, developed by Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani.

The advantages of LARS are:

* It is numerically efficient in contexts where p >> n (i.e., when the number of dimensions is significantly greater than the number of points)
* It is computationally just as fast as forward selection and has the same order of complexity as an ordinary least squares.
* It produces a full piecewise linear solution path, which is useful in cross-validation or similar attempts to tune the model.
* If two variables are almost equally correlated with the response, then their coefficients should increase at approximately the same rate. The algorithm thus behaves as intuition would expect, and also is more stable.
* It is easily modified to produce solutions for other estimators, like the Lasso.



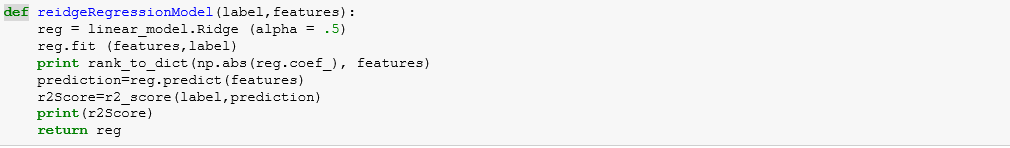


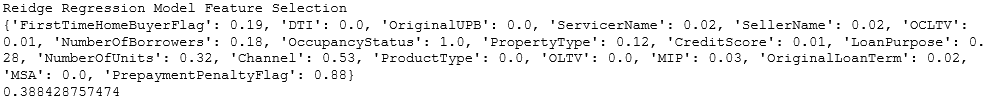
**Ridge Regression**

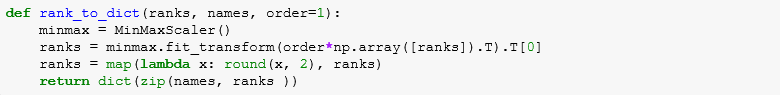
[Ridge](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge) regression addresses some of the problems of [Ordinary Least Squares](http://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares) by imposing a penalty on the size of coefficients. The ridge coefficients minimize a penalized residual sum of squares,

\underset{w}{min\,} {{|| X w - y||_2}^2 + \alpha {||w||_2}^2}

Here, \alpha \geq 0 is a complexity parameter that controls the amount of shrinkage: the larger the value of \alpha, the greater the amount of shrinkage and thus the coefficients become more robust to collinearity.







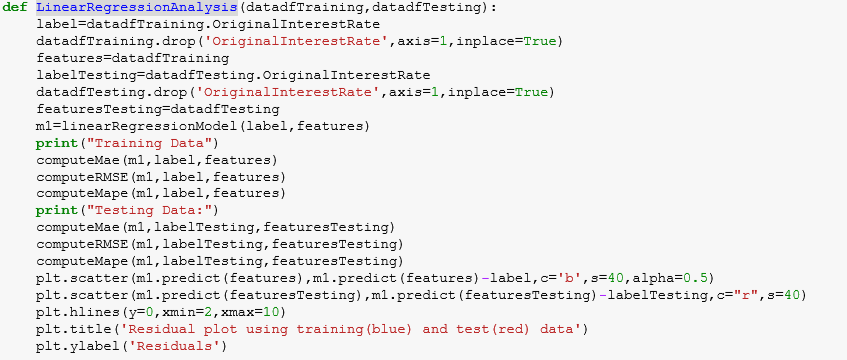
The best possible value achived for r2 score using linear regression is **0.388428757513**

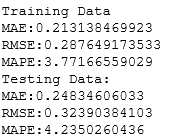
**The following columns have been used for analysis in the algorithms:**

'FirstTimeHomeBuyerFlag','DTI','OriginalUPB','ServicerName','OCLTV','NumberOfBorrowers','PropertyType','CreditScore','LoanPurpose','NumberOfUnits','Channel','ProductType','OLTV','MIP','OriginalLoanTerm','MSA,'PrepaymentPenaltyFlag'

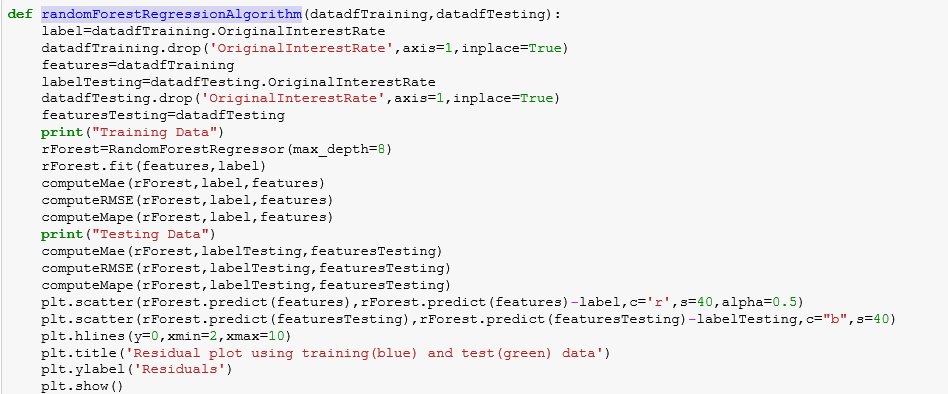
**Step 5: Machine Learning Algorithms – Supervised**

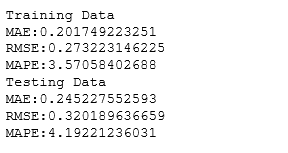
**Linear Regression Analysis**





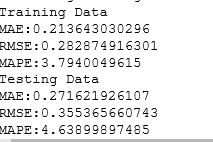
**Random Forest Regression Algorithm**





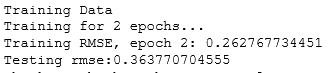
**KNN Algorithm**



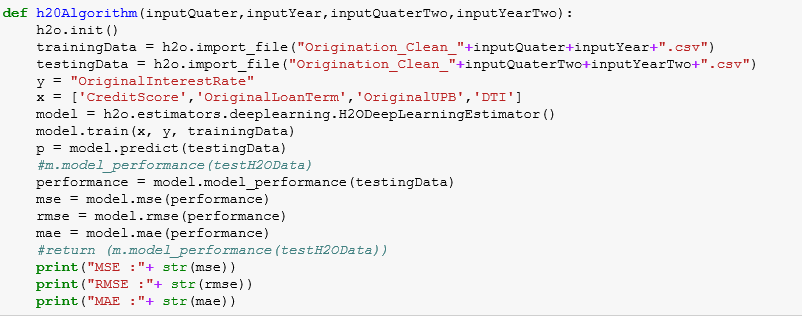


**Neural Network Algorithm - BackpropTrainer**





**H20 Algorithm**





We got the best results using **Random Forest Regression Algorithm.** Therefore we will be doing our analysis of other years using Random Forest Regression Algorithm.